MICROCOPY RESOLUTION TEST CHART
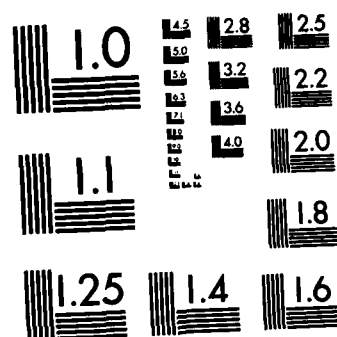NATIONAL BUREAU OF STANDARDS-1963-A

RADC-TR-82-310
Interim Report
December 1982

DISTRIBUTED DECISION MAKING ENVIRONMENT

Advanced Information & Decision Systems

J. M. Abram, C. Y. Chong, V. G. Rutenburg, E. Tse and R. P. Wishner

DTIC
ELECTE
S MAR 24 1983
D

D

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ADA126021

DTIC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-82-310 has been reviewed and is approved for publication.

APPROVED: *[signature]*
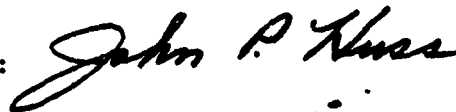
RAYMOND A. LIUZZI
Project Engineer

APPROVED: *[signature]*

JOHN J. MARCINIAK, Colonel, USAF
Chief, Command and Control Division

FOR THE COMMANDER: *[signature]*

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COTD) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-82-310	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* DISTRIBUTED DECISION MAKING ENVIRONMENT		5. TYPE OF REPORT & PERIOD COVERED Interim Report June 1981 – June 1982
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(*s*) J.M. Abram E. Tse C.Y. Chang R.P. Wishner V.G. Rutenburg		8. CONTRACT OR GRANT NUMBER(*s*) F30602-81-C-0210
9. PERFORMING ORGANIZATION NAME AND ADDRESS Advanced Information & Decision Systems 201 San Antonio Circle, Suite 286 Mountain View CA 94040		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS LDFP07C1
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COTD) Griffiss AFB NY 13441		12. REPORT DATE December 1982
		13. NUMBER OF PAGES 166
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* Same		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Dr. Raymond A. Liuzzi (COTD)

This effort was funded totally by the Laboratory Directors' Fund

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Decision Making	Resource Allocation
Design Methodology	Interactive Planning
Decentralized Control	Human Decision Making
Distributed Systems	
Distributed Artificial Intelligence	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report provides preliminary results on a distributed decision making environment. A computer implemented interactive planning environment is described, as are algorithms for automating decision making for a hypothetical Air Force scenario. A design methodology for structuring distributed decision making is also discussed.

DD _{1 JAN 73} FORM 1473   EDITION OF 1 NOV 65 IS OBSOLETE   UNCLASSIFIED

BLANK PAGE

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

This interim technical report describes the work that AI&DS has performed for RADC under the contract entitled "Distributed Decision Making Environment." The prime objectives of this project are to investigate and develop distributed decision making (DDM) techniques and to develop a design methodology for decision making in a distributed environment. The remainder of this section contains a brief status report and a description of the organization of this report.

1.1 PROJECT STATUS

There are several tasks to be investigated over the course of the project. To date we have emphasized what we feel to be the two most important tasks: investigating the technology of distributed decision making and developing a design methodology.

Our investigation of DDM technology began with a survey of relevant literature. The two main areas of this survey were distributed mathematical methods, including decision theory, optimization theory, and control theory; and distributed artificial intelligence.

The investigation continued with the development of several decentralized algorithms for use on a missile assignment scenario. A sequential assignment algorithm has been fully computer implemented and preliminary experiments with it have been run. An improvement on that algorithm, sequential reassignment, was developed but has not been implemented. A missile negotiation algorithm, in which automated decision makers bargain for resources, was also developed and implemented. Several other algorithms are currently in the development stage.

A related topic of research is our interactive planning simulation. Here we are developing a decision support system to aid human decision makers. The context is a distributed missile assignment problem. In this problem a distributed set of platforms are to assign their onboard missiles to targets. To date, we have developed the initial software for a single human decision maker. The next task is to enhance the useability of the system, followed by the implementation of a distributed version in which several humans can participate in experiments simultaneously. The distributed version will allow

1

us to study distributed human decision making and should lead to some new ideas for automating decision making, particularly from an artificial intelligence point of view.

We have spent some time trying to develop an AI solution to the distributed missile assignment problem. The approach that we are considering has a hierarchical decision structure, where decisions are made based on expertise stored in a knowledge base and on a set of heuristic rules. The expertise and rules will be garnered from in-house "experts" who have used the interactive planner.

As to the design methodology task, an initial desi. /as described in the project proposal. This methodology served as the model   our preliminary work on the design issue. At this point, we feel that ·   e identified the basic steps that must be performed in the design process. These steps take the user through successive levels of detail of design until the entire decision making structure has been defined. The number of levels is problem-dependent, but often three levels suffice: logical structure, physical structure, and detailed design.

We have also gained some insight into the question of how to generate structures within any given level of design. This process requires synthesis, optimization of parameters, evaluation, and selection. We believe that this is a process that should be performed by an interactive system, with a human playing the major role while being supported strongly by computer aided design tools.

1.2 REPORT ORGANIZATION

The report is organized as follows. Section 2 describes the DDM problem and the main scenario that we have worked with. Relevant literature in the areas of decentralized mathematical techniques and distributed artificial intelligence is surveyed in Section 3. In Section 4 we describe our interactive decision making simulation. Section 5 deals with a variety of automated decision making techniques that we have developed for the scenario under consideration. The design methodology is discussed in Section 6. In Section 7 we deal with the issue of human decision making. Section 8 contains conclusions. The appendix describes in great detail a marginal analysis algorithm that we have developed.

2

2. PROBLEM DESCRIPTION

2.1 PROBLEM DEFINITION

Decision making is required in a wide variety of tasks in the Air Force, including mission planning, situation assessment, air defense and logistics, to name a few. It is often necessary or desirable to divide the decision making task among several human or automated decision makers, rather than use a centralized design.

The problem addressed by this project is that of developing technology and a design methodology for distributed decision making. This development is being done in the context of Air Force tactical C^3 problems. In particular, we have focused on planning and control problems within C^3, rather than other decision making problems such as situation assessment.

A wide variety of technologies can be applied to DDM problems; these include control theory, operations research, decision theory, large-scale systems theory, artificial intelligence, and decision support systems. The relevant technology also includes many distributed system architecture ideas, including node-level special-purpose hardware architectures (e.g., peripheral computers dedicated to signal processing or tracking) and communication net works and protocols. The understanding of human roles, whether individual or group, and the method and point of interfacing the human to the distributed system is an important technology issue. Finally, methods for distributing and maintaining data to support the DDM process is a relevant technology issue.

The types of distributed systems of interest are quite diverse. Both strongly distributed systems (no hierarchy) and weakly distributed systems (e.g., a geographically distributed system with a logical hierarchy imposed) are of interest. However, systems in which all decision makers are located at one node are not of interest. Thus, in the distributed system of interest we can have multiple decision processes at a node, multiple nodes at each level, and multiple levels.

Whatever the structure, each decision maker will have a problem to solve locally. A key research issue in distributed decision making is to insure that the combination of the local solutions constitutes a good (but not

3

necessarily optimal) solution to the global problem. Although the decision makers are willing to cooperate with one another for the overall good, conflicting decisions may result from different information inputs, different reasoning processes, or different local objectives. Technology must be developed to handle this problem.

The design methodology is also being developed in the context of Air Force C^3 problems. By applying the design methodology, a user will be able to generate a good DDM structure for any problem of interest. The methodology must determine the task architecture, authority architecture, information structure and distributed system architecture.

Our approach is both top-down and bottom-up. On the one hand, it is problem-oriented. Starting with a problem description we generate candidate distributed decision making systems, through the decomposition of the original problem description. At the same time, we use our understanding of the strengths and weaknesses of various DDM techniques to help choose the appropriate structure from the bottom-up. In generating a design, evaluations are performed at each structural level to select good designs. Structures can be evaluated over many attributes: the expected quality of the distributed decisions with respect to a global performance measure, reliability, computational and communications burdens, responsiveness to changing parameters, and ease of reconfiguration, for instance.

2.2 ADVANTAGES OF DISTRIBUTED APPROACHES

We introduce this section with an example to motivate the need for distributed decision making, processing, data bases, and computer architectures. The command structure of the Air Force in Europe is widely distributed geographically. Within this structure it is natural to consider a decision making aid system as consisting of (1) computers local to each command to assist with local decisions and (2) communication links to tie the computers into a network to assist in data transmission for higher level (hierarchical) or cooperative (heterarchical) decisions. When one considers that Soviet doctrine towards the European theater calls for quickly cutting all NATO communication lines in Western Europe, the need is clear for keeping decision making systems (humans and machines) as close as possible to the resources they impact (sensors and weapons). Also clear is the need for a

4

highly survivable communications backbone through the use of redundant paths linking the distributed computers.

The remainder of this section discusses the many advantages of distribution: fit to problem, reliability, modularity, throughput and response time, and cost.

2.2.1 Matching the Architecture to the Problem

As pointed out above, a distributed system is often a good match to the physical characteristics of a problem, especially when the multiple foci of interest are geographically widespread. Within a single node or cluster of nodes within the distributed network (a location), a number of benefits may accrue. This location may operate in relative independence from the network, which results in reduced system complexity. The less complex local subsystem will be easier to tune for best performance under the specific local conditions. In most cases, data can be stored and processing done locally. This is one factor in the higher system throughput and reliability discussed later. Finally, in complex group decision making, the data and processing that are unique to each location contribute to the diversity of viewpoints within the entire system that is important in many decision tasks.

2.2.2 Reliability

Keeping locally that data which is used locally promotes local survivability or system-wide robustness if another location is not working. If a data base is spread throughout the network, then the risk is spread also. If one location is out, a distributed process may still be able to function with reduced capability.

The redundancy made feasible by the use of identical, mass-produced processing and communication components increases reliability and maintain-ability. It also reduces costs and promotes expandability and dynamic load-sharing and routing of communications.

2.2.3 Modularity

The modularity due to replicated components supports system flexibility including expandability and reconfigurability.

2.2.4 Throughput and Response Time

Throughput is increased by providing faster response to local requests, by taking advantage of the asynchronous parallelism possible in working on different problems in different parts of the network at the same time, and by allowing underutilized parts of the system to help overloaded parts (load-sharing).

2.2.5 Cost

Overall cost is lowered by reducing communications and possibly processing, by using mass-produced components, and by using the less expensive components available with today's technology, such as microprocessors, sensors and custom VLSI chips.

2.3 A PLANNING AND CONTROL SCENARIO

The issues to be investigated for this project are numerous. To enable us to focus on these issues, we decided that it was necessary to choose a specific scenario to study initially, rather than consider the entire class of planning and control problems. We intend to generalize the results of our investigation later on by considering other scenarios.

Our current scenario is in the area of futuristic air offense. Mission planning is to be done for aircraft and for air-launched missiles, for example cruise missiles. Additional friendly entities could include ground bases, processing centers, or satellites. The objective is to plan a coordinated attack against the set of enemy targets, which are protected by a set of enemy defensive units. Targets are assumed to have explicit values associated with them, defenses have only implicit value in that attacking them may make it easier to destroy targets, and both targets and defenses are assumed stationary.

The planning problem consists of making a number of interrelated decisions: trajectories must be chosen for the aircraft, each missile to be used in the mission must be assigned to a defense or target, launch points and trajectories must be chosen for each of those missiles, etc. Issues that enter into the planning include finding the trajectories that maximize probability of survival while satisfying constraints, deciding how many

6

missiles to use against defenses and how many to save for the targets, timing individual flights so that if a defense is to be attacked then no missiles or aircraft should fly past that defense until after it is attacked, and the trade-off between flying the launch platforms in closer to the targets to give the missiles greater effective range and threatening the platform and its missiles by therefore exposing the platform to greater threat from defensive sites.

Even from a centralized point of view, these are interesting issues to investigate. However, we are looking at these issues within the DDM frame work. Several DDM structures are possible within this scenario. One could use single-level distributed control with the decisions made either at the ground bases, aboard the launch platforms, or aboard the missiles themselves (strictly automated on the missiles, of course), allowing communications between the decision makers. Hierarchical control structures also present themselves. For example, ground bases could allocate launcher and missile resources to enemy sectors, launchers could choose their own trajectories, and missiles could select their own targets and trajectories. Communication channels would exist within and between the different decision making levels to allow for coordination. A more detailed description of the scenario can be found in Section 4.1.

3. SURVEY OF RELEVANT TECHNOLOGY

3.1 DECENTRALIZED MATHEMATICAL METHODS

3.1.1 Introduction

Decision making is an activity that all human beings engage in. Thus it is a subject which has been studied by researchers in many diverse disciplines such as economics, management science, engineering, medicine and psychology, etc. Many different theories and approaches to decision making exist. For example, prescriptive mathematical theories of decision making originated with the theory of games of von Neumann and Morgenstern [1], and the statistical decision theory of Fisher [2], Wald [3] and Savage [4]. At the same time psychologists have been investigating experimental situations for the study of decision behavior [5]- [8]. Recently, a hybrid "behavioral decision theory," which combines psychological investigations with mathematical theory, has evolved. Reviews of this area can be found in [9,10], and the book [11].

The study of distributed decision making is more recent, although the key assumption of game theory is the existence of multiple decision makers with different objectives. Most of the work in distributed decision making has appeared in the literature of system and control theory (see [12]- [14] for earlier surveys), decision and organization thoery [15] and optimization theory [16]- [18]. Related work can be found in the literature of distributed artificial intelligence [19]. However, to the best of our knowledge, there is not much of a "distributed behavioral decision theory." In this memo, we survey the literature on distributed decision making in the areas of decision theory, optimization techniques and control theory. These three areas are chosen to be surveyed together because they are all prescriptive in nature. Philosophically, they rely on models and objective functions and focus on the problem of optimizing the objective functions given the models and the constraints. They are different in the kind of models used and the specific solution techniques. However, there are sufficient similarities for them to be considered together. This is especially true in the distributed situation when some of the techniques have their origins in multiple disciplines. The subject of distributed artificial intelligence is surveyed in Section 3.2.

8

The structure of this section is as follows. In Section 3.1.2, we present the main issues in distributed decision making to facilitate the discussion and comparison of various approaches. In Sections 3.1.3 to 3.1.5 we survey distributed decision making in the three approaches. Each section is introduced with a discussion of the centralized framework and is followed by a description of the existing results for the distributed case.

3.1.2 Issues in Distributed Decision Making

3.1.2.1 Introduction

A decision problem consists of the following main ingredients: a decision maker or a group of decision makers and the environment. The decision makers make decisions which, when implemented, influence the environment. The decisions are made using the current available information which can be on-line (real time) or off-line (a priori). If the decision problem is dynamic the information at any time may also depend on the decisions made and implemented at an earlier instant (see Fig. 3-1).
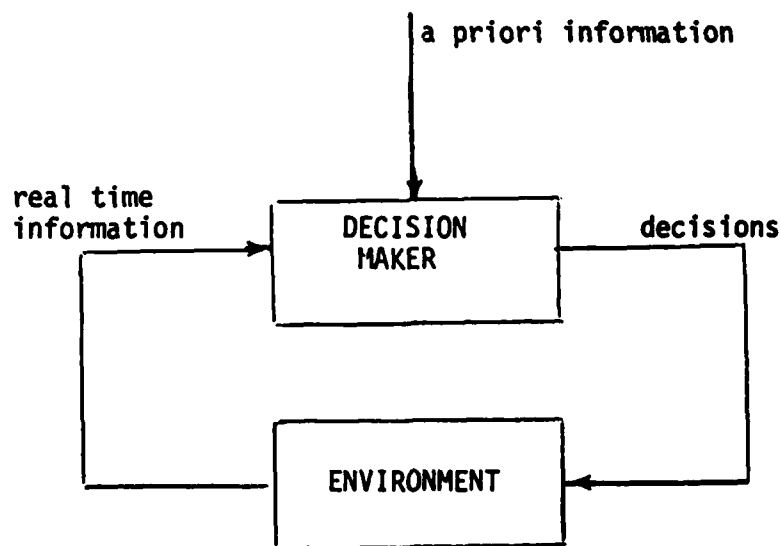
Figure 3-1 Dynamic Decision Making

A decision process generally involves several steps: problem recognition and structuring, options (alternatives) generation, decision selection and implementation. As shown in Figure 3-2, it is frequently necessary to go through many iterations between the steps before a decision is made and implemented. Most of the work in the decision making literature, and almost all the work in the areas we survey, deals with the decision selection process. Thus, in the context of this survey, decision making is the choice of an appropriate action from a set of alternatives given the available information.

3.1.2.2 Advantages of Distributed Decision Making

There are different reasons for adopting a distributed decision making structure over a centralized structure. Sometimes, distributed decision making may be the only option when a centralized structure is not feasible. At other times, the centralized option exists, but distributed decision making may be preferred. The advantages usually given for distributed decision making are:

1. Better performance, such as: faster response since the decision makers are closer to the environment; increased reliability since there is not a central decision maker whose failure may cause havoc to the system.

2. More efficient utilization of communication and/or computation resources.

3. Ease of design since the design of one central decision unit is replaced by the design of smaller units.

4. Better growth potential.

Not all of these advantages are present in every distributed decision making system. In the following survey we discuss the different approaches with respect to presence or absence of these advantages.

3.1.2.3 What is Distributed?

As discussed in Section 3.1.1, in this survey we are particularly interested in the problem of selecting an appropriate action from a set of alternatives given the available information. In distributed decision making,
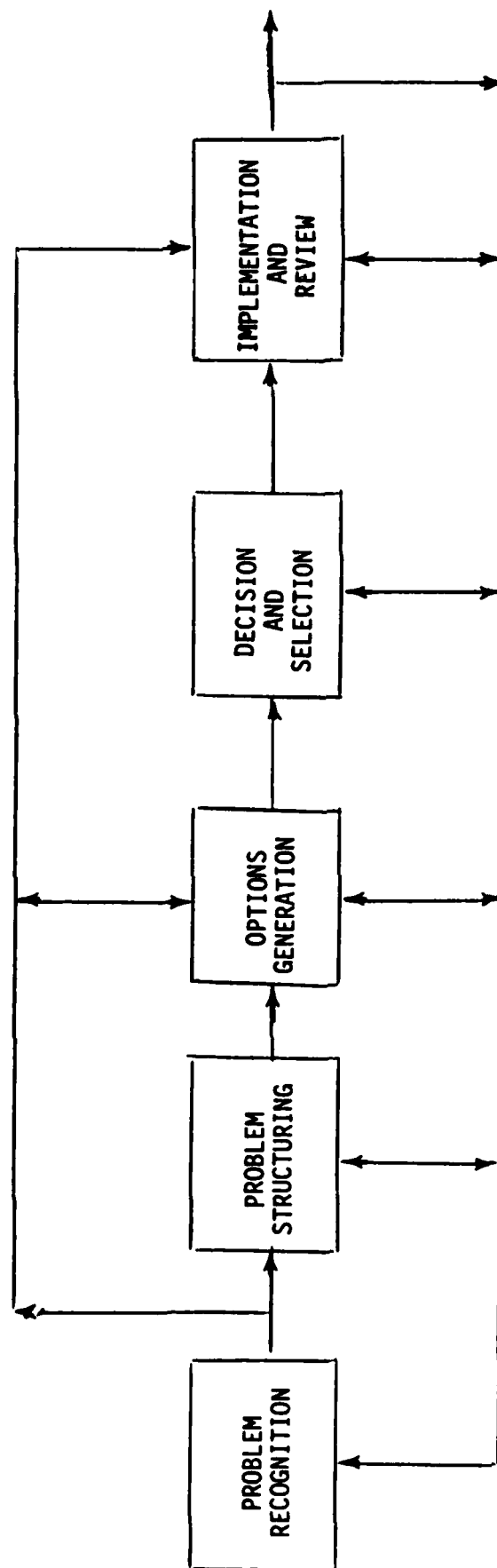
Figure 3-2 The Decision-Making Process

11

either the information (off-line or on-line) is distributed, or the computation
involved in the selection process is distributed, or both can be distributed
at the same time. The different ways of distributing the information and
computation result in different distributed decision making algorithms.
They also have implications on which advantages discussed in Section 3.1.2.2
would actually be realized.

3.1.3 Decision Theory and Decision Analysis

3.1.3.1 Introduction

Decision theory and its more applied form, decision analysis, provide a
prescriptive methodology for decision making under uncertainty. Decision
theory has a long history dating back at least to the work of von Neumann
and Morgenstern [1], who developed the modern probabilistic theory of utility.
Statistical decision theory was established by Wald [3] and Savage [4]
who extended Wald's work by formally introducing subjective probability and
utility theory. Further work can be found in references [20] to [25].

The formalism of decision theory includes the following basic ingredients.
A decision maker has to pick an action (u) from a set of alternatives. The
consequence of his choice depends not only on the action itself but also on
the state of the world (x) which is not known exactly to the decision maker.
Any choice function for actions will thus depend on two components: the
evaluation of the consequences and the relative strength of belief in the
occurrence of the different states of the world. In decision theory, the
individual decision maker's ordering of consequences is represented by a
numerical function called the utility function L(u,x) which reflects his
risk preference. The ranking of the likelihood of alternative states is
represented by the subjective probabilities p(x). The utility function (see
[26] for a survey of utility theory) and the subjective probabilities represent
the basic preferences and judgments of the decision maker. Once these
are specified, decision theory states that a rational decision maker would
maximize the expected utility to obtain the optimal decision, i.e.,

$$\underset{u}{\text{Max }} E[L(u,x)] \qquad\qquad (3.1)$$

where E denotes the expectation operator given the subjective probabilities on x and

$$E[L(u,x)] = \int L(u,x)p(x)dx$$
$$\text{or } \sum_{i} L(u,x_i)p(x_i), \qquad (3.2)$$

depending on whether x is continuous or discrete.

In the formulation of (3.1), the decision problem is static in nature, i.e., u is chosen before the state of the world is observed. This is frequently expressed in terms of a decision tree.

□ decision (act) fork   o chance (event) fork
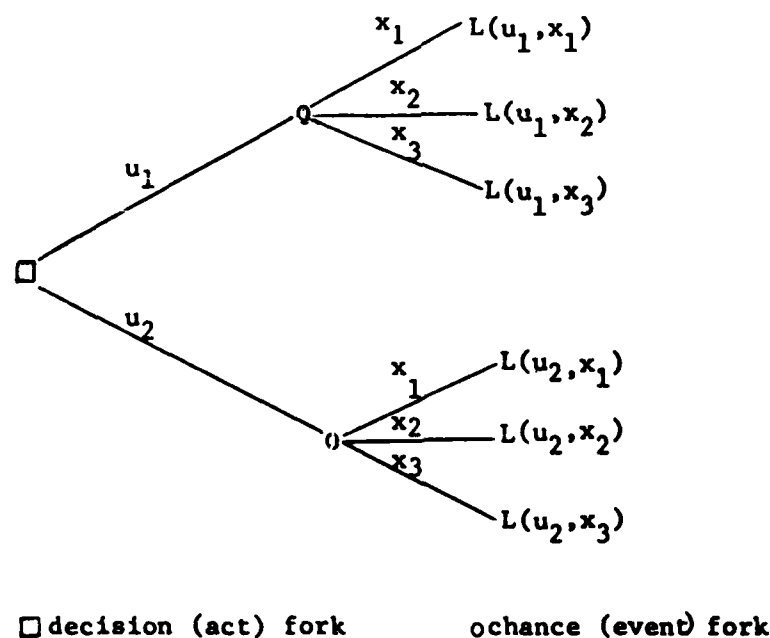
Figure 3-3   Decision Tree

In Figure 3-3, there are two alternative actions and three possible states of the work. The optimal decision is found by maximizing $E[L(u_i,x)]$. In general, however, decision problems may involve a sequence of decisions, each to be made after some uncertain outcomes are observed. Graphically, the decision tree is now as given in Figure 3-4.
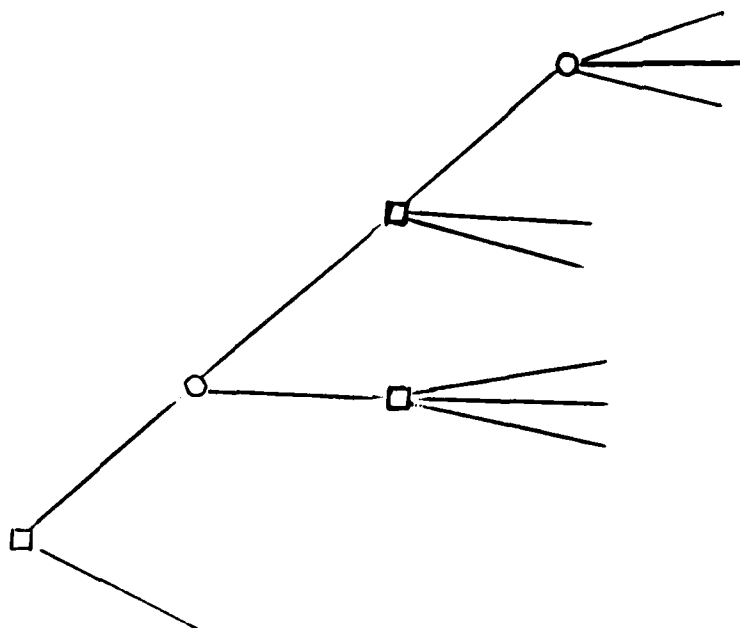
Figure 3-4   Sequential Decision Problem

Mathematically, both the state and the decision are now vectors partitioned as $x = (x_1....x_N)$, $u = (u_1....u_N)$ and u_i is allowed to depend on a subset of x. The problem is solved by starting at the final outcome (the very tips of the tree) and working back by the successive use of two steps:

1) an expectation process at each chance fork

2) a choice process that selects the path yielding the maximum future evaluation at each decision fork.

This averaging out and folding back procedure is actually backward dynamic programming.

Using decision theory as the mathematical foundation, decision analysis [27,28] provides a quantitative method for the systematic evaluation of alternative actions. It entails the identification of the alternative

14

choices involved, the assignment of values (cost/benefits) for possible
consequences and the construction of the model for their interrelations.
Decision analysts are particularly concerned with the most practical problems
of eliciting the subjective probabilities from the experts and encoding the
preferences of the decision makers. Using this information, the probable loss
or gain associated with alternative choices can be analyzed.

Much of the work in decision analysis has been performed at two private
contract research firms, SRI [29] and Decisions and Design, Inc. [30]. It has
been addressed to military problems [30] as well as other applications,
from seeding hurricanes [31] to power system planning [32].

3.1.3.2 Team Decision Theory

Classical decision theory and decision analysis are centralized, i.e.,
there is only one decision maker with a utility function and a set of subjective
probabilities about the state of the world. The presence of multiple decision
makers makes the problem more complicated because the decision makers may
have different utility functions and/or different assumptions about the
underlying uncertainty. Even when all the decision makers have the same view
of the world and are going to make their decisions cooperatively, there still
remains the problem of defining optimality for multiple utility functions.
One approach is given by multiattribute utility theory [33] where an
organizational utility function is constructed from the individual utilities.
When this approach is not valid the multiple utility nature of the problem
has to be considered explicitly and each decision maker has to model the impact
of the other decision makers on his own utility function. This problem is
studied in game theory [1] for which a large body of literature exists. Game
theorists are concerned mainly with a normative theory on the behaviors of the
individuals in a gaming situation. Various solution concepts for the multiple
objectives of the decision makers have been defined [20], dealing with both
cooperative and noncooperative situations. However, game theory is more useful
as a language and as an explanation of behavior than as a prescriptive theory for
making decisions. Another approach is given by team decision theory which

15

considers decision making by multiple decision makers with a single common objective but different information about the underlying uncertainty. Physically, one may imagine the decision makers to be connected by a communication network which is imperfect.

Team decision problems were first introduced by Marschak to study management organizations in 1955 [34] and are still studied by economists [35]-[37]. Radner presented some of the first analytical results [38]. A lot of these earlier results can be found in [39]. An early application was given for the airline reservation problem [40]. Some of the original motivations for the development of the theory have been largely eliminated through the general availability of computer information networks, e.g., airline reservation systems. However, problems where information is generated and used in a distributed manner still arise and fall naturally into the realm of team decision theory.

The following is a general model for team-theoretic decision problems [41,42] and can be viewed as a natural extension of the classical centralized decision problem. There are five principal ingredients:

1. A vector of random variables $x = [x_1 \ldots x_m]$ with a given probability density function $p(x)$. This represents all the uncertainties that are relevant to the problem, including random disturbance, observation uncertainty, etc. x is frequently referred to as the <u>state</u> of nature and lies in a possible set X.

2. A set of n <u>observations</u> $z = [z_1 \ldots z_n]$ which are given functions of x, i.e.,

$$z_i = \eta_i(x_1 \ldots x_m) \qquad i = 1, \ldots, n. \qquad (3.3)$$

$z_i \epsilon Z$ is the <u>observation</u> available to the ith decision maker. The set $\{\eta_i | i=1 \ldots, n\}$ is called the <u>information structure</u> of the problem Note that the subjective probability of the ith decision maker is now $p(x|z_i)$, which is in general different from that of the other decision makers.

3. A set of decision variables, $u = [u_1 \ldots u_n]$, one per decision maker. Each u_i <u>lies in a possible</u> set of decision alternatives U_i.

16

4. The <u>strategy</u> (decision rule, control law) of the ith decision
   maker is a map $\gamma_i: Z_i \rightarrow U_i$, i.e., the decision of the ith
   decision maker can only depend on the information z_i.

$$u_i = \gamma_i(z_i) \qquad (3.4)$$

where γ_i is to be chosen from a set of admissible strategies Γ_i

5. The <u>loss</u> or utility function of the problem is a map L: X x U → R,
   i.e.,

$$\text{Loss} = L(u_1 \ldots u_n, x)$$

Frequently, the loss or utility function is defined indirectly in
terms of an outcome or a consequence which then depends on u and x.
Note that the loss function depends on the state x and the decisions
of all the decision makers u.

For any decision strategy, $\gamma = (\gamma_1, \ldots, \gamma_n)$ the decision u becomes a
random variable and thus L becomes a random variable. The expectation of L
can then be taken with respect to $p(x)$:

$$J(\gamma) \stackrel{\Delta}{=} E[L(\gamma_1(z_1), \gamma_2(z_2) \ldots \gamma_n(z_n), x)].$$

The team decision problem is then

$$\underset{\gamma \epsilon \Gamma}{\text{Min}} \quad J(\gamma), \qquad (3.5)$$

or find the optimal decision strategy γ^* which minimizes the expected loss.
Alternatively one can maximize the expected utility.

Much of the research in team decision theory is concerned with the
investigation of optimal decision strategies for various assumptions on x,y,γ and
L. There are two main classes of teams: static teams and dynamic teams.
These will be discussed separately.

3.1.3.2.1 <u>Static Teams</u>

A static team is one where the observations of the decision makers
depend only on the state x, as in the formulation above. Conceptually the
solution of the problem is quite straightforward. For linear information

17

structures (η_i), Gaussian random states (x) and a quadratic loss (utility) function (L), the optimal decision strategies can be shown to be linear [38], ie., γ_i is of the form

$$\gamma_i(z_i) = A_i z_i + b_i.$$ (3.6)

The matrices A_i and vectors b_i $i=1...n$ can be computed easily from the problem parameters but the computation is centralized. When these simplifying assumptions no longer hold, the solution becomes more complicated but is still conceptually tractable. In essence, the problem is still that of finding a set of decision rules γ_i^*, $i=1,....,n$ such that for each i

$$\text{Arg} \min_{u_i} E[L(\gamma_i^*(z_1),...\gamma_{i-1}^*(z_{i-1}),u_i,\gamma_{i+1}^*(z_{i+1})....x)|z_i]$$ (3.7)

is consistent with γ_i^*. Here Arg Min denotes the value of u_i that minimizes u_i the function. Intuitively, equation (3.7) states that given the optimal decision rules of the other decision makers, decision maker i is faced with the usual decision theoretic problem where the subjective probability is now $p(x|z_i)$ and the utility function is expressed as

$$L(\gamma_1^*(\eta_1(x))....,\gamma_{i-1}^*(\eta_{i-1}(x)),u_i,\gamma_{i+1}^*(\eta_{i+1}(x))...,x),$$ (3.8)

i.e., it depends on x through the observations and decision rules of the other decision makers. Conceivably an iterative scheme can be devised to solve for the optimal decision rules. One can start with a guess of the set $\gamma_1,....\gamma_n$, and solve equation (3.7) for each i. If the solutions are consistent with the original set of γ's, then the set is optimal. Otherwise, the γ_i's should be adjusted until the consistency conditions of (3.7) are satisfied.

3.1.3.2.2 Dynamic Teams

If in equation (3.3), z_i depends on u_j, $j=1,...n$ in addition to x, i.e.,

$$z_i = \gamma_i(u_1,...,u_n,x),$$ (3.9)

then the information structure is dynamic and the problem becomes a <u>dynamic team</u>.

In order for the problem to be well-posed, certain causality constraints, such as a decision cannot depend on an observation which depends on that decision, have to be satisfied [44]. Intuitively an information structure is dynamic if the information of a decision maker depends not only on the state of the world but also on the decisions of the decision makers preceding him.

Certain dynamic information structures are equivalent to static information structures. The broadest class is that of _partially nested information structures_. In a partially nested information structure, if z_i, the observation of the ith decision maker, depends on the decisions u_j, $j \in N_i$, then the observations z_j, $j \in N_i$ are all contained in z_i. More succinctly, if u_k affects z_i, then the ith decision maker knows what the kth decision maker knows. It can be shown that a partially nested information structure is equivalent to a static information structure since the ith decision maker can eliminate the effects of u_j, $j \in N_i$ from his knowledge of z_j, $j \in N_i$. Most of the analytically soluble team decision problems which have appeared in the literature have partially nested information structures. An example is when each decision maker has perfect recall of all his past observations and decisions.

When the information structure of the team is not partially nested, the solution becomes very difficult. This is due to the fact that we are dealing with an optimization problem in the $\gamma_1, \gamma_2 \ldots \gamma_n$ space and there is no simple way of solving this functional minimization problem. Also, if u_j can modify the observation z_i received by the ith decision maker, then decision maker j can encode his own observation z_j in u_j and reduces the ith decision maker's uncertainty about the state of the world. The effect, called _signalling_ [44]-[46] is a major contributor to the complexity of the problem since the optimization procedure has to carry out a trade-off study between signalling and direct optimization of the expected loss.

Dynamic teams are closely related to the decentralized stochastic control of dynamic systems. Thus we will consider this area in more detail again when we survey the literature in decentralized stochastic control.

3.1.3.3 Summary

Team theory deals mainly with the situation where the multiple decision makers cannot communicate their observations efficiently in real time, so that the on-line information used in their decisions is decentralized. The decision rules, however, are still computed in a centralized manner by all the decision makers. Thus, referring to our discussion in Section 3.1.2, team theory is mostly concerned with satisfying the communication constraints or conserving the communication resources. Given these communication requirements, the performance is to be optimized. Since the real time information is decentralized, the system is quite reliable with respect to communication failure. Also, the absence of a central real time processing agent increases its reliability. However, since the computation of the decision strategies is carried out at a central site, the overall decision system is probably not very adaptive to changes in the system configuration. Thus team theory is most applicable when decentralization of the real time information is the overriding concern.

So far, most of the results in team theory are still theoretical in nature. One may speculate whether team theory will form the theoretical foundation for a methodology in distributed decision analysis just as decision theory leads to decision analysis. The answer is probably that substantial theoretical advances have to be made before this can happen. Decision theory is based on the application of dynamic programming and Bayesian analysis, which are very well understood. In a situation where information is decentralized both of these tools are no longer applicable and their distributed versions still remain to be discovered. Thus, except for simple single-stage problems, not much can be said about distributed decision analysis.

20

3.1.4 Optimization Theory

3.1.4.1 Introduction

Optimization theory is a very broad area, spanning static as well as dynamic problems. Broadly defined, it includes both decision theory and optimal control, although these areas have special additional features which set them apart from straightforward optimization. For example, the presence of uncertainty is essential in statistical decision theory while control theory deals primarily with dynamical systems. When applied to decision making, optimization is used to select the optimal decision from a set of alternatives. The existence of a well-defined objective function and a set of feasible alternatives have to be assumed. Common optimization techniques include linear and nonlinear programming, integer programming, network flow techniques and dynamic programming.

When the dimension of the problem is large, it is desirable and sometimes essential to distribute the computation involved so that it can be solved by a number of processing units (decision makers), each handling a small problem. Since Dantzig-Wolfe's decomposition algorithm [47] in linear programming and the Arrow-Hurwicz paper [48] on decentralized resource allocation, there has been a lot of work on distributed optimization algorithms [16]-[18]. There are two major classes of algorithms: those where the decision makers are arranged in a hierarchy and those where they are truly distributed with no particular center. We shall consider both classes separately.

3.1.4.2 Multilevel (Hierarchical) Algorithms

Multilevel algorithms are based on the philosophy of decomposition and coordination. Typically, the original large-scale mathematical programming problem is manipulated into a form which lends itself readily to decomposition. These decomposed problems are then solved independently by the local decision makers. Usually, however, the original problem is not completely decomposable. Thus a coordinator is needed to influence the problems of the local decision makers so that the global solution is obtained. The lower level decision

makers and the coordinator form a hierarchy as shown in Fig. 3-5. The local
decision makers solve the decomposed problems independently given some coordina-
tion parameters sent from the coordinator. Information about the solutions
of the local problems is sent to the coordinator who then adjusts the coordination
parameters. The process is then repeated until no further improvements can
be made.
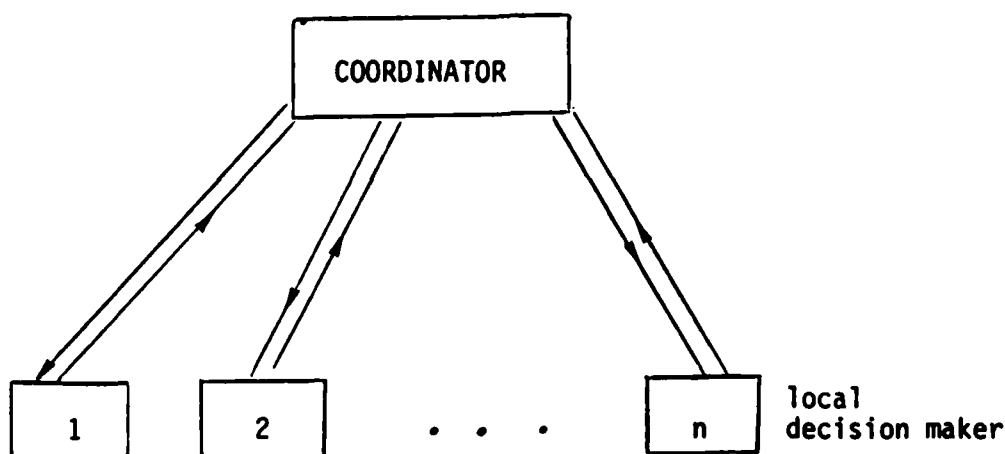
Figure 3-5 Multilevel Optimization

There are two main ways of problem manipulation [16] to obtain the
decomposition. One is primal; the other is dual. The primal methods include
separation, projection, inner and outer linearizations. After the so called
master problem is obtained, solution strategies including piecewise techniques,
restriction and relaxation are used. An example of a primal variable method is
the following [16].

The given optimization problem is:

$$\underset{x}{\text{Max}} \quad \sum_{i=1}^{n} f_i(x_i) \qquad\qquad (3.10)$$

subject to
$$h_i(x_i) \geq 0 \qquad i=1,\ldots,n$$

$$\sum_{i=1}^{n} g_i(x_i) = b$$

where f_i, h_i and g_i are all concave differentiable functions of the vector x_i.

22

First the problem is projected onto the space of its coupling constraints by introducing the vectors $y_1, \ldots, y_n$. Then we have

$$\text{Max}_x \sum_{i=1}^{n} f_i(x_i) \tag{3.11}$$

subject to

$$h_i(x_i) \geq 0, \quad g_i(x_i) = y_i \quad i=1,\ldots,n$$

$$\sum_{i=1}^{n} y_i = b .$$

If the y_i's are given, then (3.11) separates into n independent problems of the form

$$\text{Max}_{x_i} f_i(x_i) \tag{3.12}$$

subject to

$$h_i(x_i) \geq 0 \quad g_i(x_i) = y_i .$$

Each of these is the problem solved by the ith decision maker at the lower level. Let $v_i(y_i)$ be the maximum of the ith local problem given y_i. The coordinator's problem is then

$$\text{Max}_y \sum_{i=1}^{n} v_i(y_i) \tag{3.13}$$

subject to

$$\sum_{i=1}^{n} y_i = b.$$

The feasible directions strategy for (3.13) is to generate an improving sequence of y's. Note that y_i can be regarded as the allowable resource for the ith local decision maker. Thus in the primal method, the coordinator actually allocates the resources.

If we dualize (3.10), we obtain

$$\text{Max}_x \sum_{i=1}^{n} f_i(x_i) + \lambda [b - \sum_{i=1}^{n} g(x_i)] \tag{3.14}$$

23

or

$$\text{Max}_{x} \quad \sum_{i=1}^{n} (f_i(x_i) - \lambda g(x_i)) + \lambda b . \tag{3.15}$$

Once λ is chosen, the inidividual decision makers can solve their problems independently :

$$\text{Max}_{x_i} \quad f(x_i) - \lambda g(x_i)$$

subject to

$$h_{\pm}(x_i) \geqslant 0 \tag{3.16}$$

The coordinator's job is to adjust λ until

$$b = \sum_{i=1}^{n} g(x_i) \tag{3.17}$$

One possibility is to adjust $\lambda(t)$ according to the rule

$$\dot{\lambda}(t) = \alpha (\sum_{i=1}^{n} g(x_i(t)) - b). \tag{3.18}$$

In this algorithm, the price of the resource at time t coordinates the n decision makers who optimize, given $\lambda(t)$. The excess demand is used by the coordinator to modify the price $\lambda(t)$. This is in essence the Arrow-Hurwitz algorithm [48].

Multilevel algorithms have always been the principal way of decomposing a large-scale mathematical programming problem. Dantzig-Wolfe's algorithm [47] is a two-level algorithm, so is Lasdon et al's technique [49] on separable= mathematical programming. The work of Cohen [50] shows that there is a close relationship between the well-known classical techniques (gradient, Newton-Raphson, etc.) and most of the two-level algorithms. This may account for their abundance.

The primary reason for using a hierarchical approach is to distribute the computation involved in a large optimization problem among the decision makers in the hierarchy. Since computation time usually rises at a faster than linear rate with problem size, the solution of the many smaller problems will be more efficient than the solution of the original large problem.

Moreover, since the lower-level problems are independent of one another given the coordinating parameters, parallel computation is possible. However, the coordinator needs to supply the coordinating parameters in an iterative manner. If the number of interactions is large, the computational savings of this multilevel approach may be lost. It is difficult to make general judgements as to when efficient coordination is possible. One situation where decomposition helps is when the number of interconnection equations is small. Some estimates of computation time for a simple case can be found in [51].

Some information decentralization is also achieved with the multilevel approach. The coordinator does not need to know the global problem and each local decision maker needs to know only his subproblem (objective and constraints). A lot of information transfer, however, is needed between the coordinator and the lower-level decision makers. Eventually the equivalent of enough communication to transfer the global problem may have been made.

A multilevel system is often claimed to be more reliable than a centralized system. This is true if the failures occur with the lower-level decision makers. Then the coordinator can modify the coordination algorithm to adapt to the new structure. If the coordinator or the communication network linking it to the lower decision makers fails, then the performance of the system can be severely affected. Thus the claim that a multi-level system is more reliable should be qualified. In the next section we shall consider truly distributed algorithms which would be more reliable.

3.1.4.3 Distributed Algorithms

Truly distributed optimization algorithms have been studied in two closely related areas: dynamic programming and network flow problems. These are considered separately.

3.1.4.3.1 Dynamic Programming

A. Spatial Dynamic Programming

Dynamic programming [52] is a decomposition method for handling large-scale optimization problems. The application of dynamic programming embeds an optimization problem with special structure into a sequence

25

of smaller optimization problems, which are presumably easier to solve. Usually the decomposition is with respect to time or stage. Thus it is particularly suitable for optimization of dynamic systems where the temporal structure is already present. It can be shown [53] that temporal decomposition is not the only way of decomposing a mathematical optimization problem by means of dynamic programming. When the spatial structure of a system consisting of subsystems is exploited, the result is spatial dynamic programming [54],[55]. In spatial dynamic programming, the stage variable now corresponds to each subsystem, and the state corresponds to the interaction variables. The optimal solution parametrized on the possible interaction variables is found for each subsystem. The partially optimal solutions are then successively combined to produce the globally optimal solution. For example, if we consider the problem given by equation (3.10), each subsystem will solve the problem given by equation (3.12) parametrized on y_i to obtain the optimal solution $v_i(y_i)$. These functions $v_i(y_i)$ are then passed on to the other decision makers, consecutively from 1 to n. Each decision maker solves for all $0 \leqslant z_i \leqslant b$, the problem

$$\underset{0 \leqslant y_i \leqslant z_i}{\text{Max}} \quad v_i(z_i - y_i) + J_{i-1}(y_i) \qquad (3.19)$$

to obtain $J_i(z_i)$. The optimal solution is then obtained by the nth decision maker with $z_n = b$. The optimal interaction variables are found in a backward sweep from the nth decision maker to the first. An application of this idea to resource allocation can be found in [56].

In spatial dynamic programming, each decision maker has to optimize his subproblem for all possible values of the interaction variables. Unless an analytical solution can be found, this would have to be performed numerically given specific values of the interaction variables. Thus, even though the amount of computation for each optimization may be small, the total amount over all interaction values may be large. Furthermore, these optimal solutions have to be transmitted to the other subsystems. In general, the application of spatial dynamic programming is only suitable

when the interaction variables are few in number and can be parametrized efficiently. Otherwise, even though the algorithm is noniterative compared to the multilevel approach, the amount of computation or communication involved may be overwhelming.

The spatial dynamic programming approach is quite reliable with respect to subsystem failure. The algorithm itself is modular in structure, as long as there is a local decision maker to take care of the optimization. When subsystems are added or removed, the overall system can be reoptimized without the need of solving all the subsystem optimization problems over again.

B. Distributed Dynamic Programming

Another way of distributing the computation involved in dynamic programming has been proposed by Bertsekas [57]. There are straightforward schemes of distributing the calculations involved in each recursion of the usual dynamic programming algorithm among several processors. These schemes usually require that all processors must complete their assigned portions of the computation before a new stage in the recursion can begin. Thus processor synchronization is necessary and the speed of computation may be limited by the slowest processor. The distributed dynamic programming algorithm of [57] considers asychronous distribution of computation for a class of dynamic programming problems including shortest path problems (which probably provided the original motivation) as well as finite and infinite time horizon stochastic optimal control problems.

The basic formuation of the dynamic programming problem considered is given below. Let x and u be the state and control belonging to the spaces S and C respectively. For each $x \epsilon S$, $V(x) \subset C$ is the control constraint set. Let F be the set of all extended real valued functions on S. A partial order on F is defined as

$$J_1 \leq J_2 \quad \text{if } J_1(x) \leq J_2(x) \qquad \forall\, x \epsilon S$$

$$J_1 = J_2 \quad \text{if } J_1(x) = J_2(x) \qquad \forall\, x \epsilon S. \qquad (3.20)$$

Let H be a monotone extended real-valued function on S x C x F, i.e.,

$$H(x,u,J_1) < H(x,u,J_2) \quad \forall J_1 < J_2 .$$ (3.21)

H gives the optimal cost starting from x if control u is used. The
dynamic programming problem is then to find a function $J* \epsilon \overline{F}$ where $\overline{F}$ is
a given subset of F such that $\forall x \epsilon S$

$$J*(x) = \inf_{u \epsilon U(x)} H(x,u,J*) .$$ (3.22)

$J*(x)$ is then the optimal cost-to-go starting from the state x and
equation (3.22) is the standard Bellman's equation in dynamic program-
ming. One can then view the solution of the dynamic programming problem as
finding a fixed point of the mapping T within $\overline{F}$, where $T:F \rightarrow F$ is defined by

$$T(J)(x) = \inf_{u \epsilon U(x)} H(x,u,J) .$$ (3.23)

It is assumed that n computer centers (decision makers) or nodes
are present. The state space S is partitioned into n disjoint subsets
denoted by $S_1,...,S_n$. Each node i has the responsibility of finding J*
for all states x in the set S_i. Each node has a set of neighbors whose
estimates of J are needed for the computation of T(J) at the ith node. At
each time, node i can be in one of three possible states: compute,
transmit or idle. In the compute state, each node i computes a new
estimate of the values of the solution function J* for all states $x \epsilon S_i$,
using the estimated values of the function obtained from its neighbors.
In the transmit state, node i communicates the estimate obtained from
the latest compute phase to other nodes. In the idle state, node i
does nothing related to the solution of the problem. Under certain
assumptions on the problem structure, and the timing and ordering of
the computation and internode communication, the algorithm converges to
the optimum.

28

This algorithm is essentially iterative, as compared to the sequential dynamic programming. It requires little computation at each stage and the communication is restricted to between neighbors. The number of iterations required to produce a solution may be large, however. Since there is no center or coordinator involved, the algorithm should be reliable for certain kinds of node failure, at least when the nodes can discover the source of the failure and reinitialize the algorithm. This, however, is not a problem whose solution is obvious.

When applied to the shortest path problem, this distributed algorithm yields essentially the original routing algorithm implemented in the ARPANET [58].

3.1.4.3.2 Network Problems

The nice mathematical structure of many network problems provides a natural setting for the use of distributed algorithms. Lau, Persiano and Varaiya [59] developed a decentralized algorithm for solving the problem of maximum flow through a network. Decentralized schemes for achieving a common steady-state frequency for a set of spatially separated oscillators have been considered in [60] and [61].

By far the most successful application of distributed algorithms has been in the routing problem in packet-switched (or store-and-forward) computer communication networks [58], [62] to [77]. Apart from the algorithm developed by Cantor and Gerla [62], which is basically static and hierarchical in nature, all the other algorithms are truly distributed and deal with quasi-static routing.

Distributed algorithms are naturally of interest in quasi-static routing. Quasi-static routing problems are typified by the situation where a network is in operation but changes in the routes are needed because of changing traffic patterns such as the establishment/termination of communication

sessions or the addition/elimination of links. A centralized approach to the problem where one node periodically collects information from all the other nodes about the traffic requirements and uses this information to solve the current static routing problem is not practical for the following reasons. First, protocols are needed for communicating information and decisions between the central node and all the other nodes. Second, if there is a failure in the network, centralized routing suffers from the "chicken and egg" problem of needing routes to communicate the network information to establish new routes. Thus distributed routing algorithms such as those used in the ARPANET [58], [63], [64] are essential.

There are two main classes of routing algorithms. In single path algorithms, such as the ones used in the ARPANET, each packet is sent over a route to minimize the delay to its destination with no regard to other packet's delays. In multipath algorithms given the total incoming traffic for various destinations, each node in the network attempts to find the fractions of the total traffic to be sent along various paths from that node so as to minimize the overall delay for all messages. The two approaches will be considered separately in the sequel.

A. Single Path Routing

Single-path routing problems are really shortest path problems. Distributed shortest path problems have been considered by Abram [65], Abram and Rhodes [66] and by Merlin and Segall [67]. The shortest path problem can be formulated as follows:

Consider a directed graph consisting of n nodes A = {1,...,n} and a collection of links L = {(i,j):i,j A and there exists a link from i to j}. To each link (i,j) is associated a distance $d(i,j)$ which in routing problems corresponds to the estimated delay between node i and node j. It is desired to find the shortest path (with respect to the distance d) from any arbitrary node to any other node.

It is fairly obvious that this problem can be solved by centralized dynamic programming and possibly by distributed dynamic programming [57]. Abram and

30

Rhodes [66] considered decentralized shortest path algorithms that enable
each node in a network to compute its shortest distance to any other node
using only local knowledge of the network topology and only local information
transfer between adjacent nodes. In their static algorithm, which is similar
to those found in [58],[68] and [69], each node keeps an assessment of the
current estimated shortest distance via each of its neighbors. Based on these
distances the current shortest distance to the destination as well as the
neighbor(s) via which this minimal distance is achieved can be found.
Whenever a node's current shortest distance to a destination decreases, either
through initialization or new information received from a neighbor, the node
transmits this distance to all its neighbors, who then update the corresponding
distance assessments by adding this distance to the known lengths of the links to
the sender. The algorithm continues in this manner and is shown to converge
in finite time to the solution of the shortest path problem.

Their dynamic algorithm handles dynamic networks in which branch lengths
may decrease or increase, new branches (nodes) may be introduced into the
network and branches (nodes) may be removed from the network. After a
change in the network has occurred, initialization procedures are used to
insure that a node only accepts information from nodes which are aware of
the changes in the network.

Merlin and Segall [67] considered a distributed routing algorithm which
adapts to changes in the network flow requirements and converges to the
shortest paths in finite time. The key feature of their algorithm is that
it provides for finite time recovery of the routes after arbitrary topological
changes in the network, such as link (node) outages or link (node) additions.
Both of the algorithms mentioned can handle topological changes. There are
substantial differences between them. For example in [66], the node detecting
the topological change generates the reinitialization; in [67], each
distination controls the updating of routing paths. This implies some
different information requirements.

B. Multipath Routing

Multipath algorithms, as typified by those considered by Gallager [70], can be regarded as solutions to the following convex multicommodity network flow problem in the routing variables $\phi_{ik}(j)$, $j = 1\ldots,n$, $(i,i)\epsilon L$:

$$\text{Minimize} \qquad \sum_{(i,k)\epsilon L} D_{ik}(f_{ik}) \qquad\qquad (3.24)$$

subject to

$$\left. \begin{array}{l} t_i(j) = r_i(j) + \sum_k t_j(j)\phi_{ki}(j) \\[2mm] f_{ik} = \sum_j t_i(j)\phi_{ik}(j) \\[2mm] \phi_{ik}(j) \geq 0 \quad \text{for all } (i,k)\epsilon L, \; j = 1,\ldots,n \end{array} \right\} \qquad (3.25)$$

where D_{ik} are convex functions representing the delay on link (i.k), $r_i(j)$ is the expected traffic entering the node i and destined for node j, $t_i(j)$ is the total expected traffic (node flow) at node i destined for node j, $\phi_{ik}(j)$ is the fraction of the node flow $t_i(j)$ routed over link (i,k) and f_{ik} is the expected traffic on link (i,k).

Obviously each node i should decrease those routing variables $\phi_{ik}(j)$ for which the marginal delay is large and increase those for which it is small. Gallager's algorithm consists of two parts: a protocol between nodes to calculate the marginal delays and an algorithm for modifying the routing variable. Both of these can be implemented in a distributed fashion, requiring only communication between neighbors. For stationary input traffic statistics, the delay through the network coverges, with successive updates of the routing variables, to the minimum average delay over all routing assignments.

Second derivative versions of Gallager's algorithm have been considered in [71],[72] and have been shown to possess nice convergence properties. Other distributed multiple path routing algorithms have been considered in [73], where the price (goal) coordination technique of large-scale system optimization is used. While this normally results in a hierarchical structure with a

coordinator, as pointed out in the previous section, in this particular case the function of the coordinator can be distributed among the nodes in the network. Implementation of the algorithm will then only require information exchange between the adjacent nodes.

All the distributed network algorithms satisfy the decentralized information and computation requirements. For single path problems, because of the discrete nature of the problem, finite time convergence of the algorithm is often attained. In multiple path problems, the continuous nature of the algorithm implies that there is no finite time convergence. Thus to achieve any reasonable performance, a large number of iterations may be needed for some problems.

3.1.4.4 Summary

We have considered both multilevel and distributed optimization algorithms. Except for spatial dynamic programming, whose applicability to practical problems still remains to be established, most of the algorithms are iterative in nature, requiring communication between the decision makers. There is, however, some decentralization in both information and computation.

The main advantage of multilevel algorithms is its ease of design. Well established steps exist for converting a wide class of optimization problems into a multilevel structure. Resource allocation problems belong to this class. By contrast, finding a truly distributed algorithm for a problem may be more difficult. For example, many multilevel algorithms for resource allocation exist ([48],[78], and others), but there are not many truly distributed algorithms [79],[80]. Distributed algorithms seem to be more applicable to network structures, but even in this case there is no systematic way of developing the algorithms.

The main disadvantage of multilevel algorithms is the presence of a center or coordinator. When a failure occurs in the coordinator, the performance degradation can be substantial. Thus a distributed algorithm is generally more reliable.

33

3.1.5 Control Theory

3.1.5.1 Introduction

Control theory is primarily concerned with the feedback control of dynamical processes (systems). Given a dynamical system (where the output at any time depends not only on the instantaneous input but may depend on the past input history), it is desired to design a control input so that the performance of the system is satisfactory. If the control depends on the real time measurements, then it is a closed loop control; otherwise it is open-loop. The structure of a closed loop control system is shown in Figure 3-6. Control engineers rely very heavily on the use of the models of the dynamic processes. Thus over the years, tools have been developed for modeling and representation of dynamic systems, for describing their qualitative properties such as stability, and for designing controllers given these models. Though model simplification methods and stability tests for large-scale systems are relevant to decentralized control [13], our survey is concerned primarily with the distributed algorithms for controlling a system. We distinguish between three types of methods: decentralized stochastic control, hierarchical control and multi-model control.

3.1.5.2 Decentralized Stochastic Control

Traditional centralized control presumes the structure shown in Figure 3-6, where the controller can use all the measurements to generate the control. In decentralized control, there are restrictions on the information flow between measurements and controls. For example, in Figure 3-7, control 1 can use only the measurement 1, control 2 can use measurement 2, and so on. Two classes of decentralized control problems have been studied in the literature. The first class is deterministic and is concerned with the design of decentralized controllers to stabilize the resulting system. Since this is primarily an extension of classical servomechanism control, rather than distributed decision making, it will not be considered here. The second class is decentralized stochastic control, which is the subject of our current survey.
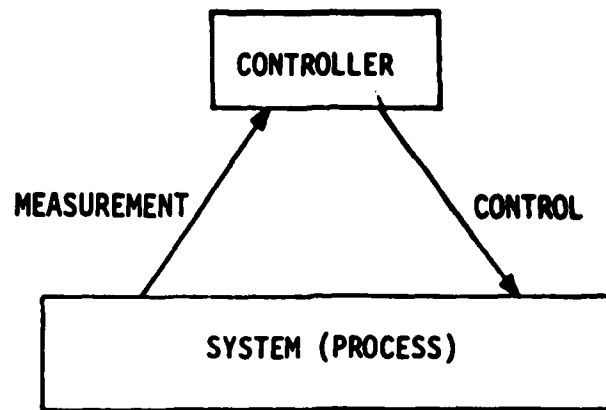
Figure 3-6 Centralized Control
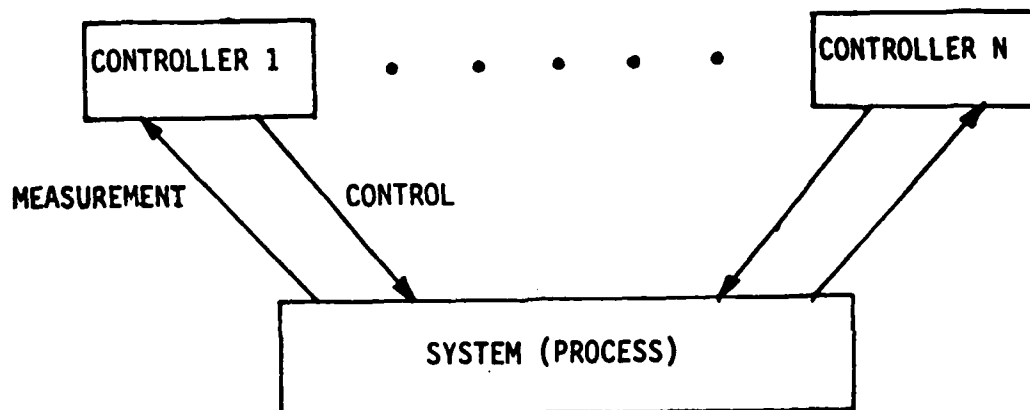
Figure 3-7 Decentralized Control

35

Stochastic optimal control deals with the following problem (in discrete time with time index k)

Given a dynamic system

$$x_{k+1} = f(x_k, u_k, w_k) \qquad \text{state equation} \qquad (3.26)$$

with measurement

$$y_k = g(x_k, u_k) \qquad \text{output equation} \qquad (3.27)$$

find a control law $u_k = \gamma(y_k, y_{k-1}, \dots)$ to optimize a scalar performance index. Here x_k, u_k, y_k, w_k, v_k represent the state, input, measurement, driving and measurement noises respectively at time k. Dynamic programming is usually used to solve this problem (see, for example [81]). With minor restrictions on the performance index and the nature of the noises, the optimal control law is known to possess the separation property, i.e., it can be synthesized by combining an optimal state estimator with an optimal deterministic controller. When the state and measurement equations are linear, the noises are Gaussian and the performance index is a quadratic function of the state and control variables (LQG assumption), the state estimator is the Kalman filter, and the optimal deterministic controller is also linear.

This problem bears some resemblance to the team decision problem discussed before. Indeed, by identifying the control action at each time with a different decision maker, we have a dynamic team problem with the partially nested information structure. For an LQG problem, linearity of the optimal control law follows immediately (see Section 3.1.3.2).

In a multi-controller version of optimal stochastic control, the output equation is replaced by n output equations relating the measurements y_{ki}, $i=1,\dots,n$ of the n decision makers to the state x and the measurement noises v_{ki}:

$$y_{ki} = g_i(x_k, v_{ki}) . \qquad (3.28)$$

Let $Y_k = \{y_{ji}; j=0,\dots,k; i=1,\dots,n\}$ be the complete set of past measurements available at time k. Each decision maker's control u_{ki} is only allowed to depend on a subset of Y_k, i.e.,

36

$$u_{ki} = \gamma_{ki}(Z_{ki})\ldots \tag{3.29}$$

where Z_{ki} is a subset of Y_k. The Z_{ki}'s specify the portions of the total measurements available to the controllers and form the information structure (pattern) of the problem [41],[82]. When $Z_{ki} \neq Y_k$, the information pattern is said to be nonclassical, and the controllers have to select their controls on the basis of different information. This is equivalent to the dynamic team problem previously discussed. When the equivalent dynamic team problem has a partially nested information structure, the solution of the linear quadratic Gaussian (LQG) problem still has a nice structure. The so-called one-step-delay sharing information structure falls into this class. This is the situation where the controllers share all their information except the most recent measurements [45],[83].

When the information structure is nonclassical and not partially nested, the solution of the optimal stochastic control problem becomes very difficult. Witsenhausen's famous counterexample [84] shows that for a simple LQG problem, a nonlinear control law can outperform the best linear law, enen though in the centralized case, the optimal control law is linear. The optimal solution to Witsenhausen's problem is still unknown. The nonlinear nature of the optimal solution is due to the fact that the control now serves a dual purpose: direct control of the system and communication to the other controllers through the system dynamics. This is the effect of signalling that we referred to earlier in Section 3.1.3.2. In a decentralized control system, it is thus possible to use the system under control as some kind of implicit communication channel, which the controllers can use for signalling. When the information structure is partially nested, a controller already knows everything that can affect its own measurement, and thus there is no need for signalling.

Various approaches to get around this difficulty have been considered in the literature to obtain more practical control stategies at the expense of decreased optimality. The most obvious one is to restrict the class of control strategies to be linear. This approach runs into a difficulty due to the so-called second guessing phenomenon, [85],[86], which arises in the following way: Each controller estimates the state of the system and the state estimates of the other controllers who in turn do exactly the same thing to estimate these estimates. This implies that the controllers are infinite-dimensional

37

and require a lot of memory. Another approach is to require the controllers to have a fixed structure and optimize within this class. This has been used in [85],[87]. Conceptually, the problem is now reduced to one of parameter optimization.

Applications of decentralized stochastic optimal control have been found in dynamic file assignment in computer networks [88], and control of multi-access satellite broadcast channels [89]-[93]. This problem falls very naturally within decentralized stochastic control since the users are usually remote from another, and due to the nonzero propagation and transmission delays for the signals to go from one station to the other stations via the satellite, the information structure is naturally decentralized.

Decentralized stochastic control can be viewed as a special case of dynamic team theory applied to dynamic systems. Thus this approach possesses all the advantages and disadvantages of the team-theoretic approach discussed in Section 3.1.3.3. In particular, its emphasis is the decentralization of the real time information, but the computation of the control strategies is centralized.

3.1.5.3 Hierarchical Control

The theory of controlling multilevel hierarchical systems was introduced by Mesarovic, Macko and Takahara [84] around 1970. Since then, important advances have been made [95]-[97]. For surveys in this area, see [51],[98]. This theory is inspired by the decomposition techniques in large optimization problems and is based on the following ideas. Many large systems are actually interconnected systems, i.e., collections of subsystems coupled together by interaction variables, and their objectives often are separable functions (or sums) of subsystem objectives. If the interaction variables are fixed, or completely free, then the system decomposes into completely independent subsystems. The decomposition suggests a natural structure for control, whereby each subsystem is controlled by a separate "lower-level" or "infimal" controller based on a simplified subsystem model and local objectives. The interaction between the subsystems is taken care of by an "upper-level" or "supremal" controller who coordinates the lower-level units by manipulating their models or objectives. The overall system then has the structure in Figure 3-8.
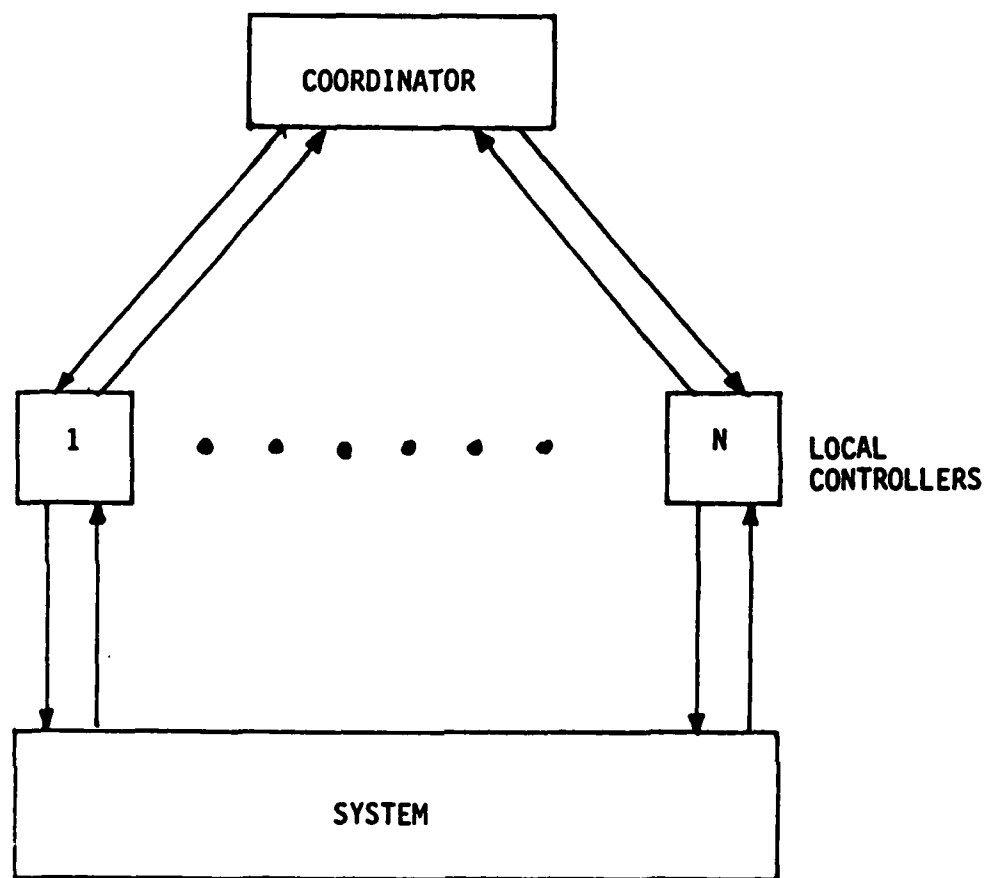
Figure 3-8 Two-level Hierarchical Control System

The kind of system under study is usually assumed to be of (or can be manipulated into) the following form. There are n dynamical subsystems represented by differential equations,

$$\dot{x}_i(t) = f_i(x_i, u_i, t) + v_i(t) \qquad i=1,\ldots,n \; , \qquad (3.30)$$

where x_i is the state, u_i is the control, and v_i is the interaction input. The interaction input $v_i(t)$ is related to the subsystem outputs by

$$v_i(t) = \sum_j g_{ij}(x_j, t) \qquad\qquad (3.31)$$

In addition, there may be local constraints of the form

$$q_i(x_i, u_i, t) \leq 0. \qquad\qquad (3.32)$$

The objective of control is to minimize

$$\int_{t_0}^{t_1} c(x(t), u(t), t)dt \qquad\qquad (3.33)$$

where

$$c(x, u, t) = \sum_i c_i(x_i, u_i, t)$$

subject to the above constraints and possibly some constraints on $u_i(t)$.

The first step in applying hierarchical control is decomposition, where the original global problem is replaced by a family of n decoupled subproblems parametrized by $\alpha = (\alpha_1, \ldots, \alpha_n)$. It is necessary for the coordinability condition to be satisfied, i.e., there exists a parameter such that the solution of the decoupled problems yields the solution of the global problem. The second step is an iterative procedure whereby the local controllers solve their independent infimal problems using the coordinating parameters from the coordinator and send the results of their solutions to the coordinator, who then readjusts the coordinating parameters. This process is then repeated until a satisfactory solution is obtained.

There are two main approaches to decomposition and coordination in hierarchical control. The first approach, called interaction prediction in [94] and the direct method in [97], is as follows: The coordinator chooses the interaction input $v = (v_I, \ldots, v_n)$ and transmits it to the lower level. The lower level problem then decomposes into

$$\underset{u_i}{\text{Min}} \int_{t_0}^{t_1} c_i(x_i, u_i, t)dt \overset{\Delta}{=} J_i(v_i) \tag{3.34}$$

subject to

$$\dot{x}_i = f_i(x_i, u_i, t) + v_i(t), \tag{3.35}$$

$$q_i(x_i, u_i, t) \leq 0. \tag{3.36}$$

The upper level problem is

$$\underset{v}{\text{Min}} \sum_i J_i(v_i) \tag{3.37}$$

which is equivalent to ensuring that

$$v_i(t) = \sum_j g_{ij}(x_j, t). \tag{3.38}$$

Usually a gradient method is used to solve the upper-level problem. Two difficulties may arise in using this approach. First, the v chosen by the upper level may be such that the lower-level problems are infeasible. Second, a gradient procedure for solving the upper-level problem may not be possible because of the nondifferentiability of the functional involved.

The second approach is called interaction balance or the dual method. Here the coordinator modifies the local objectives by dualizing the original problem. With the price vector as the coordinating parameter α, the lower level problem decomposes into

$$\underset{u_i, v_i}{\text{Min}} \int_{t_0}^{t_1} c_i(x_i, u_i, t) + \alpha_i^T v_i - \sum_j \alpha_j g_{ji}(z_i, t)dt$$

subject to the same constraints (3.35) and (3.36). Note that the controls at the lower level now include both u_i and v_i. The aim of the coordinator is then to select $\alpha*$ such that the coupling constraint is satisfied. For

applicability, this type of coordination usually requires some convexity condition on the objective functions and linear coupling constraints.

Because of its origins in mathematical programming, hierarchical control is applicable mostly to deterministic problems and for off-line solution of open-loop controls. An extension to on-line hierarchical control can be found in [99]. The periodic coordination approach of Chong and Athans [100] represents one of the few attempts to merge hierarchical system theory and multi-person stochastic control. In this work, the coordinator supervises the activities of the lower-level system control and every now and then influences the lower-level controller to:

(1) Set it straight - correct the estimates generated by the lower level; these estimates are incorrect because the lower-level controllers only have local information.

(2) Change its directives - generate new targets for the lower level.

(3) Change its incentives - so that the lower-level objectives can be modified to conform with the global objectives.

There is a class of games which have some relevance in heirarchical control: Stackelberg games are those where one player is called the leader and the others are followers. The followers pick their strategies to optimize their local objectives given the actions of the leader. The leader then chooses his strategy to optimize his own objective. It can be observed that this structure is very similar to the hierarchical structure introduced in this section if we identify the leader with the coordinator and the followers with the lower level decision makers. The research on the Stackelberg approach to multilevel systems [101],[102], however, is concerned more with finding the optimal strategies in a centralized manner.

Except for the variations mentioned in the last two paragraphs, hierarchical control is essentially multilevel optimization theory applied to dynamic systems. All the comments made on the advantages and disadvantages of multilevel optimization theory also apply here.

3.1.5.4 Multimodel Control

If decentralized real time information of the decision makers is important, decentralized stochastic control has to be used. On the other hand, we have seen that the class of problems which can be solved is still quite small. The complexity of the control algorithm in decentralized stochastic control is due partly to the triple role played by the decision maker's action (control): guiding the system's behavior, "learning" about the state and signalling to other decision makers. There is no simple way to separate them, and since the problem is one of optimization, the solution becomes very complicated when all three functions have to be considered simultaneously. There have been attempts to alleviate this difficulty by introducing distributed models into decision making. Both Chong [103] and Tenney and Sandell [104],[105] recognize that a centralized model is really not suitable for decision making in a distributed system environment. It is more natural to give each decision maker a model which is compatible with his information.

For example, in an interconnected system, the decision maker for each subsystem employs a detailed model of his subsystem only and an equivalent model for the remainder of the system. Other decision makers behave similarly. Thus the overall information structure consists of decentralized measurement sets as well as decentralized model information on the same system. The multimodel aspect of the information structure is not only realistic but also eliminates the possibility of signalling in the system. This is so since the absence of a common model on the implicit channel prevents the decision makers from arriving at a signalling strategy.

Multimodelling has been studied [106] for interconnected systems strongly coupled through their slow parts and weakly coupled through their fast parts. The ith decision maker neglects the weak coupling parameters and the fast dynamics of all the other subsystems. His model is then of lower dimension than the original one and basically consists of a detailed model of his sub-system and a coarse model of the external system. Note that the model is still not well-defined until the actions of the other decision makers are specified. In [106], it is assumed that the ith decision maker knows the objectives of the other decision makers and thus the decision rules. The properties of the actual system controlled by the decision rules derived using the distributed models can then be investigated.

Distributed decision making using a distributed model has been
considered by Tenney and Sandell [104],[105]. Each decision maker is assumed
to possess a model of its own exclusive subsystem and solves its local control
problem assuming incoming interactions will be the worst possible subject to
knowledge acquired through communication. This knowledge may be expressed
either as constraints on those interactions which may occur, as cost
functions on interaction sequences, or as simplified models of the external
subsystems. The coordination schemes proposed are schemes of communication
between the decision makers to reduce the uncertainty on the interaction inputs.
This approach is influenced to a certain extent by the work in artificial
intelligence.

3.1.5.5 Summary

Of the three approaches to distributed control surveyed above, two of
these, decentralized stochastic control and hierarchical control, have their
origins in team decision theory and multilevel optimization. Thus their
relative merits and special features have already been discussed in Sections 3.1.3
and 3.1.4. The last approach of Section 3.1.5.4 tries to distribute off-line
(a priori) information as well as real time information. This is intuitively more
attractive since it is probably more realistic. Whether this approach would
be successful or not depends on the search of good coordination strategies
for which not much is known.

3.2 REVIEW OF DISTRIBUTED ARTIFICIAL INTELLIGENCE

Distributed artificial intelligence (DAI) has only recently come to be recognized as a discipline of the field of artificial intelligence (AI). Hence, this review begins with a general overview of AI and its relevance to decision making, before defining DAI and its relationship to AI and distributed systems. Then, areas of recent research in DAI are examined, including foundations; architectures; protocols, languages, and tools; hypothesis formation; and planning and control.

3.2.1 Artificial Intelligence Overview

AI (or "intelligent systems") is a largely experimental and engineering discipline engaged in an attempt to produce intelligent behavior in machines (typically including a computer). The field of AI is now over 25 years old. It grew at the intersection of the fields of software systems, theory of computation, cognitive psychology, philosophy, and cybernetics. While the field is by no means mature, a body of concepts and techniques has been developed [107] and shown useful in a number of fruitful applications [108]. These successful applications include proving mathematical theorems, solving mathematical problems symbolically, inferring the chemical structure of a compound from data gathered by scientific measurement devices, diagnosing diseases and recommending therapy in a number of areas, planning molecular genetics experiments, and geological prospecting. Applications with military relevance include image understanding for a number of types of sensors, (printed) natural language understanding, speech understanding, acoustic signal understanding, situation assessment, and mission planning.

The central notions of AI are heuristics and symbolic processing. One example of heuristics are the "rules of thumb" that guide the everyday existence of people. A heuristic, in contrast to an algorithm, may not produce an optimal solution to a problem, indeed, may not produce a solution at all. But heuristics provide a means of formalizing a way to solve a problem when more formal, guaranteed approaches don't exist or are inefficient. Symbolic processing, as opposed to numerical processing, involves the manipulation of structures of objects (e.g., words) that represent concepts. Whereas symbol processing can be either algorithmic or heuristic in nature, in general, it is clear that much of the functioning of

45

intelligence is nonnumeric in character and that symbolic processing is usually more understandable by humans than numerical processing.

Applied AI requires a system to have a large body of knowledge (the "knowledge base") about a particular problem domain in order to produce a high level of performance. Such knowledge-based systems are also "expert" systems if their heuristics have been gleaned from human experts. These performance-oriented systems are contrasted with general problem solvers based on weak heuristics, and general information processing models of human memory and cognition.

The specific areas of AI that are most relevant to decision making are knowledge representation, problem solving and inference, hypothesis formation, planning, and control of environment.

Knowledge representation is the art of selecting the method(s) by which the static information (e.g., heuristics) and dynamic information (e.g., a partial hypothesis of a scene) are to be represented in an AI system so that it will produce a good solution efficiently (or even at all). The standard forms of AI knowledge representation are predicate calculus (or formal first-order logic), production rules (antecedent-consequent rules, if-then-else rules), semantic nets (relational trees and networks), frames (templates, prototypes), computer programs (procedural embedding), and combinations of these.

Problem solving involves the inference of new knowledge (e.g., a hypothesis or planning decision from the current knowledge available to the system). Inference can be either inductive (concept formation, generalizing or learning from previous instances) or deductive (inferring specific facts about a specific situation). Inductive techniques include learning by example and reasoning by analogy. Deductive inference can be either top-down (goal-driven, backward chaining) or bottom-up (data-driven, forward chaining). Among the more important deductive techniques are formal theorem proving, problem reduction (AND/OR goal trees, divide and conquer), demons (spontaneous computation), heuristic search, generate and test, hill climbing, means-ends analysis, and constraint satisfaction.

Most AI problems may be divided into a static analysis of the

environment, planning a response, and monitoring the environment during
execution of the plan. Hypothesis formation is the interpretation or
understanding of empirical data or real-world situations. AI research has
addressed a wide range of hypothesis formation tasks, from inferring molecular
structures from mass spectroscopy data [109] to inferring goals and intentions
of military forces from their recent activities. The key problem for such
systems is dealing with the constraints imposed by the real world: missing,
incomplete, erroneous, ambiguous, and redundant data. Simple mathematical
formalisms, such as scalar certainty factors [110] have been developed for
dealing with these uncertainties and for evaluating multiple, competing
hypotheses. Richer comparison structures, such as those based on fuzzy sets
for decision making [111] and control [112], need to be introduced.

AI planning involves reasoning about a current situation, a goal, and a
set of feasible operators to produce an ordered set of operators that, when
applied to the current state of the world, will produce the goal state.

The uncertainty of how useful a plan will be when it is finally executed
motivates the area of AI control. Two approaches are to use contingency
(conditional or nonlinear) plans or to do incremental, real-time replanning.
Much of this work has been done in the area of robot arm control, but has been
done more recently for such tasks as air traffic control [113].

3.2.2 Applicability of Artificial Intelligence to Decision Making

It should be obvious at this point why AI is relevant to decision making,
especially in Air Force command and control situations, where decision making
is distributed, often done in real time, and often based on large quantities
of data. Decision making under such constraints is a highly intellectual
activity, one that might be greatly improved by some degree of automation.
Decision making involves assessing a situation, making a decision based upon
this assessment, carrying out the decision and, monitoring the execution.
These three tasks correspond closely to the AI areas of hypothesis formation,
planning, and control. In various contexts AI has squarely addressed the
problems of understanding a situation based upon sensor or nonsensor data,
integrating or fusing data of different types, choosing the best alternative
of hypotheses and plans (decisions), automating the development of AI systems
by allowing the definition of heuristics by domain experts, fostering a man-

47

machine symbiosis with interactive computing, and developing trust in the computer assistant by providing facilities to explain the machine's reasoning in an understandable way.

3.2.3 Distributed Artificial Intelligence: Definition and History

Distributed AI may be defined in a number of ways. A weak definition is that DAI encompasses any AI endeavor that utilizes a distributed computer system or architecture. This definition admits a spectrum of possible degrees of distribution and levels of communication. An AI system must have some degree of distribution of data and/or processing and/or control and some degree of communication to be considered a DAI system. Only the two extremes of the spectrum are not included: (1) a totally centralized system in which all processing takes place at one node and no external communication is required, and (2) a totally independent parallel system in which more than one node exists but no communication between nodes takes place.

A number of AI systems that are not DAI systems because of restriction 1 nevertheless were important precursors to real DAI. These systems run on a centralized computer, but are divided into a number of separate problem solving entities, typically by the different types of problem-solving know ledge present. Examples of such entities include "knowledge sources" in the HEARSAY-II speech understanding system [114] and the SIAP undersea acoustic signal understanding system [115]; "experts" in the PSI automatic programming system [116], and "beings"[117]. If the computational grain sizes (i.e., the average computing time per invocation) of the individual entities are large, they are often scheduled as independent, communicating multiple processes. HEARSAY-II was also run on a centralized multiprocessor, C.mmp [118]. However, although these systems exhibit some of the organizational and processing behavior of DAI, control and data are quite highly centralized.

The first true DAI systems, under our weak definition, are those that do problem solving at one node and merely receive data, possibly after some non-AI processing, from other nodes. Some prototype systems of this type exist, e.g., the SIAP signal understander [115] and LADDER for natural language access to a distributed database [119]. These systems introduce some of the complexities of distributed systems because of their data distribution. The issues include reliability, redundancy, missing data, communications

protocols, response time, etc. But such a system is still fundamentally a centralized AI system within a distributed processing system.

A decision making system can be described in terms of three levels of structures. The first level is data such as sensor measurements, known facts about friendly and enemy forces, knowledge about the environment, etc.

The second level is processing, i.e.,inference rules which can be applied to the available data to make further conclusions and decisions. Examples of such rules would be the Kalman filter, which estimates the target's position from the available raw measurements; the linear simplex method, that computes the optimum resource allocation from the knowledge of the linear resource constraints; and a MYCIN (IF-THEN) inference rule which determines the nature of a patient's illness on the basis of the known symptoms.

Finally, the third level,the so-called Meta-Level, is the control structure of the decision making system. This level is responsible for deciding which inference rules from the second level to apply to which portion of data, and in what sequence.

This division into the three levels is not clear cut in most cases because, for example, the dividing line between data about the world model and the inference rules about the world could be very blurred. However, practically all the existing AI methodologies and systems follow the general pattern of this three-level system.

When we talk about a distributed AI system, we need to specify the nature of the distribution present. Distribution can occur on any (or all) of the three levels of the system. Different possible combinations of distribution vs. centralization at various levels give rise to systems greatly differing in their distribution structure, whereas systems with the same distributed levels tend to have a lot of structure in common.

We now introduce our strong definition of DAI. It requires that AI problem solving take place at more than one node in the distributed network [120]. This definition forces control, data, and processing to be distributed to some degree. New and difficult AI issues must be addressed, including decomposition of problems for a distributed solution, and structures and

49

protocols for sharing and passing problem solving, control, static knowledge
bases, supporting structures for hypotheses, hypotheses themselves, plans,
etc. DAI systems of this type will use the techniques developed in the areas
of distributed processing and distributed data bases, but will necessarily
push these disciplines into new areas because of the richness of the
structures used by AI for problem solving, control and knowledge
representation.

Few viable prototype DAI systems exist today, and virtually all of these
are weak DAI systems. This is in contrast to the state of centralized AI
prototypes, which number in the hundreds. Work on strong DAI probably started
with work on a distributed problem solving protocol called contract nets
[121,122]. Research impetus for the field was generated by the question of
the role of AI in distributed sensor nets [123]. The first workshop
specifically on DAI was held less than a year ago [120]. The remainder of
this section on DAI surveys the aspects of this emerging field that are most
relevant to distributed decision making in a military command, control, and
communications environment.

3.2.4 Foundations

Designing a successful DAI system involves 1) creating local decision
making systems, 2) specifying protocols for their communication and
collaboration, and 3) deciding on the architecture for the decision making
group (i.e., which nodes communicate with each other, which, if any, nodes are
supervisors, which are subordinates, etc.).

At the present, the DAI researchers are tackling the first task by
borrowing the already existing centralized AI decision making systems (the
HEARSAY-II architecture, in particular) and using them as prototypes (with
slight modifications) for the local decision makers.

Most of the fundamental research in DAI has been devoted to the second
and the third tasks. In particular, Smith & Davis [122] have concentrated on
the design of flexible communication protocols, whereas the RAND group has
been initially more concerned with the question of logical system
decomposition, hierarchical vs. heterarchical committees and other
architectural issues. Finally, Lesser & Corkill[124-126] are attempting to

50

address both issues by creating a distributed testbed with pre-developed communication protocols and capabilities to simulate any user-specified network architecture.

Kornfeld is designing DAI systems to take advantage of the parallelism possible [127]. If a problem may be solved by a number of relatively independent techniques, they may be attempted in parallel until one succeeds. A refinement of this idea allows heuristic measures of progress to determine which approach to spend future resources on. This approach is analogous to parallel heuristic search, and may be useful in non-distributed, parallel systems as well as in DAI.

3.2.5 Architectures

A fundamental issue in DAI is system architecture, especially what types of DAI architectures are feasible and which one is best for a certain class of problems. DAI systems can be organized by both top-down decomposition and bottom-up synthesis. In fact, a combination of the two approaches is probably best. The top-down approach attempts to break a problem solving task into relatively independent subproblems that may be feasibly distributed (e.g., tasks that do not require the same hypothesis structure at every point in time). The bottom-up approach starts with either a required or proposed node and considers its relationship to other nodes to determine designs for local processing, local network topology, appropriate paradigms to use for cooperation with other nodes, etc. Possible topologies include hierarchies or trees, networks or graphs with fixed or dynamic logical substructures, linear systems, homogeneous matrices, etc.

A study has been made of a (relatively) simple distributed problem solving task, that of solving a crossword puzzle by a group [128]. Both hierarchical and heterarchical ("anarchic committee") architectures were tried. The hierarchical approach used a tree of problem solvers, each of whom was allowed to communicate only to his single superior and three subordinates (or section of the puzzle board, for the bottom row of the tree). The heterarchical approach dispensed with the two levels of management nodes and allowed horizontal communication on the main level to take place within ad hoc committees as needed. Although the heterarchical architecture gave the best performance, it isn't known when, if ever, the hierarchical approach is

better. Although the crossword puzzle problem is a fairly simple task, a number of design rules for DAI systems have been proposed. They seem to be very general guidelines, of too high a level, too vague and nonspecific, to be of great use without much refinement.

The first successful DAI system was created on the basis of HEARSAY-II[129]. HEARSAY-II is an AI hypothesis formation system (initially designed for voice recognition) with centralized control and data base but with multiple knowledge sources (decision makers) that attempt to derive important results from the data, working concurrently, and communicating their results by posting them on the global blackboard. Since all the knowledge sources had access to the blackboard, all of them were able to utilize the new results derived by any one of them as soon as this result was posted on the blackboard.

The architects of HEARSAY-II, Lesser & Erman, have since created a truly distributed AI system, i.e., a system with several control systems with local data bases working in parallel. They took three complete HEARSAY-II systems together, each with its own local blackboard, gave each one of them a partial view of the spoken message to be deciphered, and allowed them to communicate their most important (high-level) findings through one common global blackboard.

As a generalization of this effort, Lesser & Corkill[124,125,130] are presently working on a general simulation testbed for hypothesis formation. The current area of application of the testbed is in the area of street traffic monitoring. Although the application area is different, the HEARSAY-II architecture proved to be versatile enough to be still applicable. The testbed works in the following manner: The user is asked to specify the number of decision making nodes present (each one a copy of HEARSAY-II) and to specify the desired network architecture, i.e., which nodes have communication links between them, as well as the level and amount of communication allowed between the nodes. The simulation testbed then simulates the performance of this architecture, thus allowing the user to compare different plausible distributed architectures in order to choose the best ones, as well as to derive general conclusions about distributed network structures.

In order to allow more flexible forms of control among nodes in the system, Lesser & Corkill integrated goal-directed control (in the spirit of the contract net ideas developed by Smith & Davis (see below)) into the system architecture. This allows each node not only to communicate its conclusions, but also to ask other nodes for help with verifying hypotheses which are needed for making some of that node's local conclusions. This new structure of the testbed is more general and more flexible, allowing for applications in various areas of planning, as well as hypothesis formation. These applications, however, are yet to be developed, and most of the design issues are yet to be resolved.

In the area of planning and control, a number of alternative architectures for the distributed control of fleets of aircraft and cruise missiles have been proposed [131]. The best choice depends upon the requirements and constraints of the specific problem.

Davis advocates a top-down decomposition of a DAI problem into potentially independent or nearly independent subproblems [132]. Then additional internodal communication is introduced bottom-up as necessary to eliminate the chance that no solution will be found or that solutions to subproblems will be incompatible.

Hayes-Roth lists six dimensions along which problems may be distributed: space, time, instrumentality, resources, information, and interpretations [133]. In addition to these general dimensions, there may be additional, problem-specific dimensions.

3.2.6 Protocols, Languages, and Tools

The contract network [121] is one of the few attempts at a general-purpose problem solving protocol. As the name indicates, the cooperation between nodes is achieved by way of contracts between nodes. The current area of application of that system is in the area of hypothesis formation in distributed sensor networks. Each node is equipped with several sensors of various types, covering a subarea of the whole battlefield. Contracts are negotiated using the "announcement-bid-award" protocol sequence. In its efforts to track and classify the targets in its area of coverage, a node often encounters a subproblem that it cannot resolve, because of its

limitations with respect to the area of coverage, available sensor types, sensor error, etc. In such cases the node sends a "Request For Proposals" message to all possibly eligible friendly nodes.

This RFP message consists of task specification, eligibility specification, and bid specification (i.e., information requested from each bidder in order to determine the best one). If the contract needs to be fulfilled under time pressure, the expiration time for the contract would also be listed.

All the nodes that meet the eligibility specification for the task and have the available resources to tackle the task answer the RFP message with a proposal bid in which they specify their qualifications, such as the types of sensors available, the position and area of coverage of these sensors, etc.

On the basis of the bids received, the "sponsoring" node chooses the best contractor (or several sub-contractors) to do the job. This selection is announced to the chosen bidder with an award message, which specifies in detail the task to be performed. The structure of the network is democratic (or what Hayes-Roth and Wesson call anarchic), in the sense that all nodes are treated as equals, each one being a supervisor for some contracts and a contractor for some other contracts. Each contract is thus arranged on the basis of need and merit.

One of the important issues that this contract net approach will encounter will be prioritizing work on contracts. If a node has several contracts to work on in addition to its own work, prioritizing the work on all these tasks could be a real problem. A first-order solution to this problem could be achieved with the help of contract supervisor-specified priority ratings of each contract task. These ratings could change during contract work by either the supervisor or the subordinate as the circumstances change. The portion of time devoted to working on each task could then be proportional to the priority rating of that task.

The idea of contracts and negotiations also plays the main role in the cognitive science work of Fikes[134] at Xerox, who postulates that work in such cooperative domains as offices is achieved in terms of negotiating, making, and fulfilling commitments to the other workers.

As mentioned earlier, standard AI systems provide many ideas for DAI. Similarly, the succession of AI languages (e.g., QA4, QLISP, PLANNER, CONNIVER, KRL, etc.) and concurrent languages (Concurrent PASCAL, ADA, etc.) are providing the basic ideas for languages for DAI. Of particular note is the succession of languages at MIT, starting with PLANNER and ending with ACTORS. The original purpose was to provide mechanisms for goal-oriented function invocation. Recent work on the APIARY network architecture is aimed specifically at using message-oriented languages for distributed computing, including DAI [135].

Tools for DAI are obviously weak at this stage in the field's growth. The LADDER system provides a natural language front-end to a distributed data base [119]. This front end could evolve into a fairly general AI system that interprets an English request in terms of its knowledge of the network topology and what capabilities each node has. Other relevant work is now being done on AI models for effective user interfaces to distributed systems [136]. Finally, the ROSIE language and program for developing rule-based deduction systems allows a number of such systems to communicate in a distributed fashion [137]. This is the first step toward automating the construction of one class of DAI systems.

3.2.7 Hypothesis Formation

The AI area of hypothesis formation or interpretation is one approach to the Air Force problem of situation assessment. However, most AI work to date has addressed the lower-level problems of signal understanding for images, speech, and other forms.

The Rochester Intelligent Gateway (RIG) distributed computing network provides protocols for image transmission in support of distributed image understanding, both within RIG and over the ARPAnet [138]. Relaxation methods have been applied to understanding images to produce highly parallel algorithms that could be distributed [139].

The HEARSAY-II speech understander [129,140] used a distributed model of expertise, in which the hypotheses and most other data resided in a common data base, but expertise on phonemes, syllables, words, syntax, semantics, and pragmatics resided in separate knowledge sources. The distributed version of

55

HEARSAY-II was discussed earlier.

SIAP (Surveillance Integration Automation Project) is a system that was based on the HEARSAY-II architecture, but was adapted to the problem of under-sea acoustic signal fusion and understanding [115]. The key problem is the integration of signals from a distributed array of remote sensors. Another difference from the speech domain is the need for continual operation over longer periods of time than the ocean equivalent of one sentence of speech. Recently Advanced Information & Decision Systems has developed a new, strongly distributed system for situation assessment [141]. This system is possibly the first to take a combined AI and control theory approach. However, when viewed as DAI, it distributes the hypothesis formation functions of SIAP to each sensor location and relies upon nearest-neighbor communication rather than centralized communication.

The work on functionally accurate hypothesis formation by Lesser & Corkill[126] may be noted again here. Whereas distributed HEARSAY-II was purely bottom-up hypothesis formation, and use of contract nets imposes a primarily top-down orientation, the work on functionally accurate techniques recognizes the need for a mixture of both top-down and bottom-up. Perhaps SIAP provides the best balance of top-down and bottom-up processing to date: most processing was driven by the new data entering the system, yet high-level knowledge sources could also impose subgoals on lower levels. A more complete discussion of the design alternatives in AI hypothesis formation is found in [142].

The work by Wesson & Hayes-Roth[128] discussed earlier on distributed puzzle solving is aimed at providing a foundation for distributed situation assessment, but no computer system has been developed.

3.2.8 Planning and Control

Research in the area of distributed AI planning and control is under way at RAND[131]. They approached developing a theory of cooperative AI control as an empirical investigation. That is, they chose a particular area of application- control of air traffic - and intend to develop the experience and results derived from the research in this area into a general theory of distributed control. To date they have developed an initial version of a

"flexible architecture" testbed (similar in spirit to that of Lesser and
Corkill), and created six different major distribution frameworks. They are:
geography-centered distribution (based on controller's positions on the air
field); function-centered (based on types of aircraft to be controlled);
plan-centered (based on parallel problem solving ideas, similar to Kornfeld);
hierarchical (based on level of abstraction); object-centered autonomous (a
silent, autonomous structure based on aircraft self-planning) and object-
centered cooperative (a self-planning based structure in which communication
is used).

Initial simple experiments were conducted to compare performances of the
cooperative versus the autonomous architectures as a function of air traffic
density. Not surprisingly, the autonomous system performed very poorly under
high density, but equally well under moderate density. These results are what
one would expect, but they point out a hope that more meaningful results will
be derived in the future under much more elaborate experiments.

Carl Hewitt [143] proposed a mathematical notation in the form of the
PLASMA system for modeling the roles of several controllers (actors), their
effects on the environment, and their knowledge about the other actors and
their actions and methods. This system was then applied to cooperative
solving of some simple mathematical problems. Since this is a very high-level
design structure, the success of future applications of this structure to
fairly complicated situations will depend heavily on the properties of the
lower-level structures which, perhaps, need to be designed separately for each
type of application.

Work in parallel problem solving has been done by Kornfeld[144]. He
introduced the concept of "combinational implosion". "When a problem is to be
solved, attempting multiple approaches simultaneously can be advantageous
because the ultimately successful approach cannot be known in advance.
Partial information can be used to reapportion the system resources in ways
that minimize the overall expected running time." Based on Kornfeld's ideas
the Ether language has been developed at MIT for running highly parallel
programs. A variant of Ether was used for implementation of another idea of
Hewitt & Kornfeld[145], that of "Scientific Community Metaphor". It argues
that the scientific community is an excellent example of a highly successful

and highly distributed problem solving system. The problem solving paradigm that the authors believe is at work can be described as follows: When a problem is posed to the system, proposers suggest possible solutions. Proponents attempt to show the solutions work and skeptics attempt to show they will not accomplish the goal. Evaluators examine partial results as they accrue and reapportion processing power to the various approaches in ways that fit current evidence. This paradigm could provide a useful framework for some of the future distributed problem solving systems.

In the area of natural language, Allen & Small[146] are working on task-oriented dialog comprehension. Three loosely coupled levels of analysis are involved: task reasoning, communication goal reasoning and linguistic reasoning. They are now in the process of implementing their ideas. Work on planning in the highly unpredictable world of multi-agent planning has also started at MIT by Davis, et al[147]. It is also based on the negotiation paradigm.

A similar task of developing formalisms for reasoning about other agents' cognitive states and knowledge has been addressed in parallel by Appelt, and by Konolige and Nilsson[148] at SRI. Appelt's work[149] is close to the cognitive science approach taken by Allen, Cohen & Perrault[150,151] and is similar in spirit to Davis's work. A planning system called KAMP has been developed on the basis of this work. In the area of planning Konolige and Nilsson have been looking at ways of modifying the classical centralized AI systems (such as STRIPS and situation calculus) to accommodate multiple agents. This work is still in the beginning stage.

3.2.9 Summary

This section has attempted to provide a summary survey of the important work conducted in the area of distributed AI. As we have seen, there have been several attempts to search for theories of distributed AI. At the present, these theories are rather simple and high level, and much more substance is needed to make these ideas workable and applicable. At this point, the development of formalisms has not lead to a general theory for practical system construction, and the efforts in domain-specific problem solving have been more successful. This should come as no surprise, since artificial intelligence (and especially, distributed AI) is in its formative stages and, therefore, is mostly an empirical science.

4. INTERACTIVE PLANNING

As we discussed earlier, the most promising approach to the DDM problem is bottom-up, i.e., starting research with a particular scenario and then generalizing the derived results and methodology. The general planning and control scenario that we chose was discussed in Section 2.2.

In this chapter we are going to give detailed mathematical definitions of the chosen scenario and describe the rules of the computer simulation game based on this scenario (see 4.1); we will then explain the structure of the man-machine interface, discuss its planning capabilities, and give a small detailed example of how one may play this game.

This example game was played in a centralized fashion by one player. But this centralized planning mode should be viewed as a local sub-module of the distributed planning system. In Section 4.3 we will describe the structure of the distributed planning system and some of the experiments that we have run or plan to run using this system.

4.1 SCENARIO FOR DDM INTERACTIVE PLANNER

In this section we are going to describe the details of our initial DDM scenario. This scenario is a simplified model of situations encountered in modern and future real-life aircraft mission planning. However, this highly simplified model has much of the richness and complexity encountered in its real-life prototypes. Our scenario has been implemented on a computer in the form of a simulation game.

The game is played on a rectangular board, divided into square grid cells, each cell representing a land area of a certain size (say, 20 mi X 20 mi). Some of these cells are occupied by combat forces. These forces include enemy targets that we wish to attack, enemy defenses that protect approaches to the targets, and friendly missile launchers with a number of missiles on board of each of them. In Fig. 4-1 we have an illustration of such a scenario.

Initial fuel is 25

```
              T2                        T4      1  T1  100      0      T2  200    0
    T1    D2              T3                     2  T3  150      0      T4  100    0
                 D3            D5                3  D1  .99             D2  .99
    D1                D4            D6           4  D3  .99             D4  .99
                                                5  D5  .99             D6  .99
                                                6  L1  .99  Y   M1 M2
                                                7  L2  .99  Y   M3 M4
    L1                L3                         8  L3  .99  Y   M5 M6
                            L4                   9  L4  .99  Y   M7 M8
         L2                                     10
                                               11
                                               12
                                               13
    1  2  3  4  5  6  7  8  9 10 11 12 13
```

Figure 4-1   Sample Scenario

In this example there are four targets: T1 in cell (1, 1), T2 in cell
(5, 1), T3 in cell (9, 2) and T4 in cell (12, 1). These targets have
different military values assigned to them. T2 is the most valuable one: it
is worth 200 points. T3 is worth 150, while T1 and T4 are each worth 100
points. These targets are protected by six defensive SAM sites: D1 through
D6. Their threat is modeled in the following way:

The cell that the defense is located in presents immediate danger to the
aircraft and missiles flying over it; that is, there is a significant
probability that the missile (aircraft) will be shot down. Our current model
places this probability at 0.30. In addition to its own cell, the defense
also threatens the eight cells adjacent to it. In each of these cells the
defense has a slightly lower, currently 0.20, probability of shooting down the
missile that is flying there.

If a missile attacks a defense, the probability of kill is equal to the
probability that the missile survived to the defensive site multiplied by a
probability of kill constant for missiles against defenses. We are using 1.00
for that constant. Similarly, if a missile attacks a target, its probability
of kill is currently modeled as 0.80 times the probability that the missile
reaches the target. We have also assumed an additional threat that launch
platforms have a 0.01 probability of being intercepted and shot down. This
helps to prevent the decision maker from leaving missiles on launchers for
unreasonable lengths of time.

60

All evaluations are performed in a probabilistic average sense. That is, the evaluations are based entirely on expected value calculations. If a missile is shot down with probability 0.30, then it continues with a probability of existence of 0.70. We do not currently use the Monte-Carlo technique of generating a random number, deciding the outcome of the attack on the missile, and then either removing the missile or letting it continue unharmed.

There are four launchers in this scenario: L1 through L4, each carrying two missiles: M1 and M2 on L1, M3 and M4 on L2, M5 and M6 on L3, M7 and M8 on L4. The following assumptions are made:

- Each missile has a certain amount of fuel on board. In our case each missile has 25 units of fuel.

- A launcher is presumed to have a very large amount of fuel, sufficient for the completion of the whole mission.

- A launcher can fire a missile at any time.

- Each offensive piece (a launcher or a missile) can move in one of the eight basic directions.

- Each move up, down, left, or right, takes two time periods and uses up two units of fuel.

- Each diagonal move takes three time periods and three units of fuel.

- If a missile exhausts all its supply of fuel during flight, it falls on the ground and explodes in the cell it is currently over.

In order to model and address the issues of incomplete information, recognizance, contingency planning, and mid-course re-planning, there are several pop-up defensive threats placed somewhere on the board by the computer. The player does not initially know where they are. However, at the moment a missile enters the threat region of a pop-up defense, the pop-up threat becomes known and subsequently acts like any other defense. Every time a new defense pops up, the player encounters a new situation and may need to make some mid-course changes to his plans.

The object of the game is to destroy the largest expected value of the opponent's targets, i.e., to maximize the player's score. This score is equal to

$$\sum_{j=1}^{NT} \; [V_i(T_j) - V_f(T_j)]$$

where NT is the number of targets, $V_i(T_j)$ is the initial value of T_j, and $V_f(T_j)$ is the final value remaining to T_j. Other factors such as the safety of launchers, number of unused missiles, and explicit value of destroying defenses can also be incorporated into the performance measure, if desired.

4.2 THE LOCAL PLANNER

4.2.1 Functional Description

In using the planning system, the player interacts with the computer by using a set of functional commands to make decisions about the strategy to implement. The decisions that the player needs to make are:

5) Which targets to attack and which missiles to assign for each attack.

6) Which defenses to take out for easier access to the designated targets and which missiles to use for these attacks.

7) Where and when to launch each missile.

8) How to bring the launcher to the points of launching.

9) Within each individual missile-on-target or missile-on-defense attack, which path to fly so as to minimize the exposure to enemy defenses, but still operate within the fuel constraint.

If there are several pop-up threats on the field, the player should also be prepared to do some re-planning in the future, and thus should have contingency planning guidelines thought out in advance.

Examples of the commands that the player can use to execute his plans on the computer are:

1) LAunch M 3 - launch missile 3 now.

2) MOve M 5 to (7, 3) - move missile 5 to cell (7, 3) (presumed to be adjacent).

3) KIll M 4 - detonate missile 4.

4) ENd - end of the commands for the current time period, go to the next time period. (Time periods are also called update times).

5) OPtimal path from (X0, Y0) to (X1, Y1) with F units of fuel - this command asks the computer to compute the best (optimal) path from the cell (X0, Y0) to the cell (X1, Y1) within the available fuel constraint.

6) SEnd missile i along the optimal path to the target (or defense) j with specified fuel amount.

7) STay - Causes specified vehicle to remain stationary (stay) for specified length of time.

8) SKip - Skips n updates, i.e., lets the computer know that no new commands are going to be given for the next n time periods.

9) DIsplay - Displays the current position on the screen.

The commands (5) and (6) above are interesting because they presume intelligent capabilities on the part of the computer. They let the human not worry about the complicated task of finding the optimal (least dangerous) path between two points, thus allowing him to concentrate on higher-level, strategic issues. Command (6), in addition, allows the user to send out a missile (or a launcher) on a trajectory and not to worry about having to continuously guide it all the time.

The DDM game that we have implemented can serve as a basis for development of an interactive mission planning decision aid, in both its centralized and distributed versions. Since there is a strong need in the military, especially in the Air Force community, for such an aid, we are going to further develop our DDM interactive planner with an eye towards the goal of making it into a real planning asset, especially in the distributed version. The smart commands described above are the first steps towards that goal.

63

Among the new planning capabilities that we are presently implementing are:

- Capability to evaluate the goodness of a set of attack plans (assignments) before actually executing it.

- Capability to create several plans and to let the computer evaluate which plan is the most promising one.

- Capability to deal with partial plans, i.e., facility to evaluate the probabilistic outcome and consequence of a proposed new assignment command, taking into account its interdependence with the previously input commands.

- Letting the computer generate valuable suggestions with regard to creating good (high-valued) attack plans, as well as the timing of the attacks.

The above capabilities were listed basically in the order of their difficulty to implement. The first capability is a challenging task to implement, but it is known how to do this. The last task, on the other hand, is a difficult mathematical problem. In the next chapters we are going to discuss some algorithms applicable for this problem of automatic generation of optimal, or near optimal, plans.

Another set of capabilities that we are designing concern embedding the local planner into a large distributed planning system. These capabilities include:

- Capability to send high-level descriptions of intended attack plans.

- Being able to respond to and comment on other players' plans.

- Capability to evaluate the effects of other players' plans on one's own forces and plans.

- Capability to coordinate plans with other players.

4.2.2 Example of a DDM Game

In this section we shall go through an example of a DDM game. This will help the reader to understand the operation of our interactive planning

64

system. For continuity, we use the same board position as was discussed in Section 4.1:

```
Initial fuel is 25
            T2                      T4     1 T1  100      0      T2  200   0
     T1   D2          T3                   2 T3  150      0      T4  100   0
              D3           D5              3 D1  .99             D2  .99
     D1                D4           D6     4 D3  .99             D4  .99
                                           5 D5  .99             D6  .99
                                           6 L1  .99 Y  M1 M2
                                           7 L2  .99 Y  M3 M4
     L1                L3                   8 L3  .99 Y  M5 M6
                            L4              9 L4  .99 Y  M7 M8
           L2                              10
                                           11
                                           12
                                           13
     1  2  3  4  5  6  7  8  9 10 11 12 13
```

The time is 1

The scoreboard on the right has the following meaning: the first number after the symbol for targets (e.g., T3) is the current value of the target, the second number is the value already extracted from it. Thus, T3 has a current value of 150 and no value, i.e., 0 has been extracted yet. There are six defenses. The number next to the name is the current probability of existence. Four launchers are present in the battle. Each of them starts with a 0.99 probability of existence. Each launcher has two missiles on board; for example, L1 has M1 and M2, and L3 has M5 and M6. All missiles are presumed identical and all have 25 units of fuel to start with. The probabilities of kill between various objects are as specified in Section 4.1. Now, let us start playing the game. The game clock shows 1. It's our move.

First, we decide upon the plan of action. We may be tempted to send all eight missiles against the targets, say, two missiles against each. How well would we do? Let us see how easy it is to penetrate the defenses. We ask the computer to show us the best path from the spot where L1 is to target T2.

65

```
>>op path from <X Y> 2 8 to <X Y> 5 1 <fuel> 25
Optimal survival is   .45
Time to go is   17
```

```
              $                    T4    1  T1   100       0      T2   200    0
      T1      $           T3             2  T3   150       0      T4   100    0
              $    D3           D5        3  D1   .99              D2   .99
      D1      $         D4          D6    4  D3   .99              D4   .99
              $                          5  D5   .99              D6   .99
              $                          6  L1   .99  Y  M1 M2
            $                            7  L2   .99  Y  M3 M4
      $                  L3               8  L3   .99  Y  M5 M6
                           L4            9  L4   .99  Y  M7 M8
          L2                            10
                                        11
                                        12
                                        13
   1  2  3  4  5  6  7  8  9 10 11 12 13
```

As you can see, there is only a 45% probability of getting to T2 intact,
and therefore, only 36% of killing it with the first missile. The second
missile would get another 25% of the value, but even the two of them combined
will derive only 61% of the value. Similar obstacles will hinder our
performance against the other three targets.

Therefore, we need to attack some defenses first in order to create holes
in the defenses. Good candidates to attack are D2, since it controls access
to T1 and T2, and D5, since it controls access to T3 and T4. In order to
thoroughly destroy them, let us use two missiles against each, and then send
the remaining four missiles through the holes against the targets, as seen in
the following plan:

$$M_1 \rightarrow D_2$$

$$M_3 \rightarrow D_2$$

$$M_5 \rightarrow D_5$$

$$M_6 \rightarrow D_5$$

$$M_2 \rightarrow T_1$$

$$M_4 \rightarrow T_2$$

$$M_7 \rightarrow T_3$$

$$M_8 \rightarrow T_4$$

We are ready to start execution. We launch missiles M1 and M3 and send them against defense D2. We also bring launchers L1 and L2 closer to the enemy and then launch their remaining missiles (M2 and M4) when M1 and M3 are ready to hit D2. We need to keep M2 and M4 behind so that they don't get shot down by D2 before it is taken care of.

The missiles and launchers start flying. For a while, we don't have to make any new assignments. Then at time 9, when the first wave attackers get close to their chosen defenses, we launch the remaining four missiles, sending M2 against T1, M4 against T2, M7 against T3, and M8 against T4. The next interesting event occurs at time 11:

You were attacked by a pop-up near 4,4

Total score is 0

```
            T2                    T4      1  T1   100    0      T2   200    0
     T1     D2              T3            2  T3   150    0      T4   100    0
        D7        D3             D5       3  D1   .99           D2   .99
     D1    M1           D4           D6   4  D3   .99           D4   .99
           *1                 M5         5  D5   .99           D6   .99
           M4                 M6 *2      6  D7   .99
                                         7  M1   .79 t  15     M3   .99 t  15
                                         8  M5   .99 n  15     M6   .99 n  15
                                         9  M2   .91 t  23     M4   .91 t  23
                                        10  M7   .91 t  23     M8   .91 t  23
                                        11
                                        12       *1:M3 M2
                                        13       *2:M7 M8

   1  2  3  4  5  6  7  8  9 10 11 12 13
```

A pop-up threat, D7, has attacked M1, reducing its probability of existence down to 79%. This defense also threatens to keep on shooting at M1, M3 and all the following missiles along their way to D2. Thus, attacking D2 would now do very little good. D7 itself is no easy prey since it is heavily protected by D1 and D2 and since its destruction will not solve the problem of reaching T2.

Thus, we need to change our plans considerably. We divert M5 and M6 from D5 to D4. This will create a hole in the enemy defenses and will still allow M7 and M8 to fly through and attack the enemy targets. Also, it seems that M3, M2 and M4 still have enough fuel to use this hole to get at the enemy, too. After performing an analysis of the new situation, including a check on the new optimal paths from the remaining missiles to the objectives, we decide upon the following revised plan:

67

$$M_5 \to D_4$$

$$M_6 \to D_4$$

$$M_1 \to T_3$$

$$M_3 \to T_1$$

$$M_2 \to T_4$$

$$M_4 \to T_3$$

$$M_7 \to T_2$$

$$M_8 \to T_2$$

Notice how different this new plan is from the initial one. Next, we input our chosen new plan into the computer and execute it. As the execution proceeds, no new pop-up defenses are encountered, and everything goes as planned. By time 34, the mission is finished, and the final configuration is as follows:

```
Total score is 367
              T2                      T4      1  T1   67    32      T2   61   138
     T1    D2              T3                 2  T3   18   131      T4   32    67
        D7        D3               D5         3  D1  .99            D2  .99
     D1                 D4                D6   4  D3  .99            D4  .12
                                              5  D5  .99            D6  .99
                                              6  D7  .99
                                              7
                                              8
                                              9
                                             10
                                             11
                                             12
                                             13
     1  2  3  4  5  6  7  8  9 10 11 12 13
```

We have derived approximately 367 points in total, with T2 and T3 contributing the most, 138 and 131 respectively. Also, defense D4 has been effectively attacked.

4.3 DISTRIBUTED INTERACTIVE TESTBED

4.3.1 Design of DDM Testbed

In the DDM game, the role of the player is to provide command and control decisions in the form of commands like "send", "launch", "move", etc.; the role of the computer is to execute those decisions, and to provide computational support for making them. The computer also provides all the available information about the current battle, in the form of the board position and the scoreboard.

In the case of several decision makers, each commander will have only partial or even erroneous information about the battle. For example, he may know only about what is happening on the territory within his "sensor" field of view. He may also not know everything about the plans and actions of the other decision makers, due to communication constraints. All these situations can be modeled within the distributed game testbed.

This testbed consists of several modules, as seen in Figure 4-2. The central one is the Battle Simulation Module, which has the correct and complete information about the battle. This simulation module communicates with each decision maker (player) through that player's interface module. This module is very much like a version of the centralized game, discussed earlier. It acts as an interface for the player, helping him with making decisions and executing these decisions on the Battle Simulation. There are two major differences. First, the new interface does not allow the player to see all the information, but only what he is allowed to know, for example a partial view of the battle. Second, to compensate for loss of information, there are communication capabilities added to the interface. This will allow the players to communicate with each other, sharing information and plans.

The constraints on communication between the players are modeled through the use of the Communication Module. This module enforces the communication constraints by transmitting only certain types of messages or by restricting the number of messages allowed, or by imposing "taxes" on communication, or in some other way, depending on the model of communication chosen for each experiment.

69
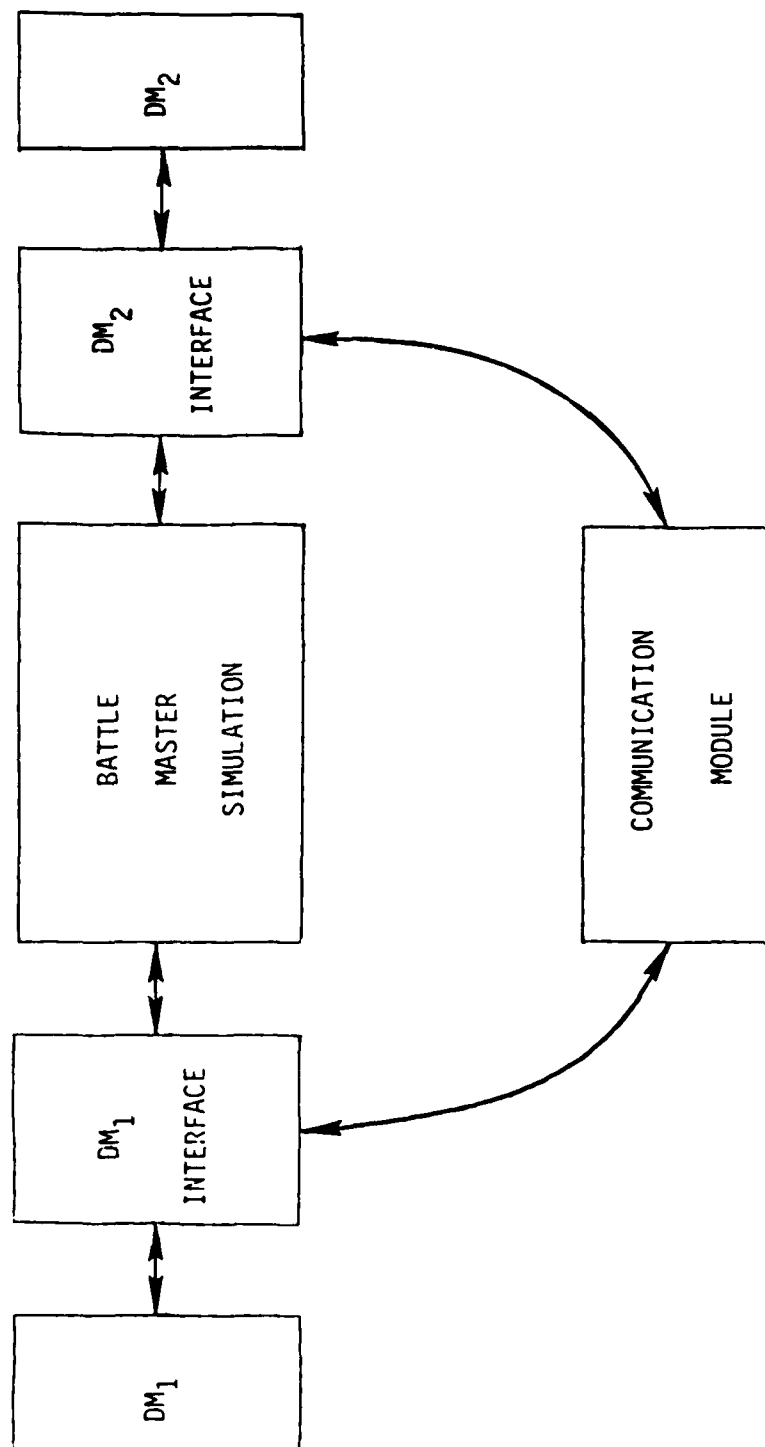
Figure 4-2 High-Level Structure of the Distributed Game Testbed (with two players)

In the next section we are going to discuss the experiments that we have run or plan to run using the distributed game testbed.

4.3.2 Design of Experiments on Distributed Mission Planning

There are three main factors that contribute to the distributed nature of mission planning:

1) Differences in information between actors. For example: each actor has first-hand information only about a part of the whole battlefield, including the enemy targets and defenses, and, especially, about the missiles and airplanes belonging to other actors

2) Difference in goals and objectives. This addresses the issue discussed in Section 7 that actors may have different objectives in their missions, each one possibly concerned about getting the most glory, or worrying about destroying only his own assigned targets, or saving his airplanes and missiles. How do you reconcile the differences between such actors? How do they reach a compromise?

3) Even if the actors all have the same goal and even if they have perfect information about the current state of the battle, there may be differences in intention. That is, Actor A could see where the airplanes of Actor B are at the moment, but he still might not know where they are flying to.

Preliminary experiments addressing the third issue have been run, although more experimenting needs to be done later. In these experiments, the players knew the current positions of all targets and missiles, but didn't tell each other their future intentions. The main issues encountered during the game were:

- Least commitment: leaving one's options open as long as possible until the other player's intentions became clear.

- Non-ambiguous actions: if one wanted the other player to understand what his mission was, one tried to execute that mission from the start in such a way that would single that mission out to the other player.

71

- If the partner's plans were fairly clear, but seemed bad, one tried various ways to indicate his displeasure, but doing that was almost impossible without direct communication.

- If one partner had bad plans, and the other couldn't indicate his disagreement, he just tried to do his best within the constraints.

- Sometimes one partner's actions seemed so puzzling that the other ignored him altogether and just acted as if he was not there.

- Avoid strategies that require really finely tuned timing.

- Use of "division of labor". It was tacitly assumed by both players that there would be "division of labor" in terms of:

a) geography: player on the left voluntarily assumes responsibility for the targets in the left field; player on the right—right field;

b) player whose forces are closer to the battle field assumes more responsibility in knocking out first line defenses, as well as in attacking "far to reach" targets;

c) player with more missiles on board was also more inclined to sacrifice some of his missiles to attack common defenses, etc.

There was another experimental distributed game run. In this game the players tried to adjust their plans before execution. Each player would decide what he wanted to do with his forces, write it down, and exchange the plans with the other player. Then the players would adjust their plans so that they would agree better with the other player's plans, and then exchange new plans again. The process continued until both players were satisfied and didn't want to exchange plans any more. The interesting issues were:

- Oscillation: Player 1 would change his plans to fit those of Player 2, Player 2 would change his plans to fit those of player 1, and they would over-shoot: another disagreement.

- Writing distorted plans in order to indicate displeasure with the other player's plans. For example, if one wanted the other player to attack Target No. 6 on his side of the battlefield, one would assign one of his

72

missiles to do that temporarily, so that the other player would realize that this needed to be done.

- Necessity to use similar types of algorithms and heuristics for all players other actors. In some of the games played the two players took different approaches to the solutions and came up with mutually inconsistent plans. Since each player was sure that his plan was the right one, there was very little yielding on either side, and it took a long and agonizing process for a compromise to be reached. This also points out a necessity for a good planning aid that would help evaluate the validity of the other player's plan.

In the near future, we plan to address issues 1) and 2) mentioned at the beginning of this section. To address issue 1), we are going to play a distributed game with two or more players, with each player knowing everything about his own resources and practically nothing about his partner's resources. He may also have only very scant information about the enemy forces outside of his area of main interest. The cost will be imposed on communication in the form of a "tax" of n points subtracted from the total score for each message transmitted. A great number of scenarios are going to be created. For each scenario, we are going to play the game with various values n of the "communication tax", from very high to very low. Then we shall compare the performances of the cooperating players for different values of the tax. Thus, it will be seen how the cost of communication influences the actions of and the nature of cooperation between the players. To compare how rational these psychological reactions to the cost of communication are, we should also run a series of the following tests:

Again, there are going to be many different scenarios. For each scenario we are going to impose several communication limits in the form of either "no more than m messages per game" or in the form of restrictions cn the types of messages allowed. There will be a wide spectrum of such limits. from very low to very high. The game will be played separately for each value of the limit. The results, i.e., the score V_0 and the total number M of messages transmitted, will be recorded. Then we shall pretend that there had been a communication tax of n points per message. Thus, the real score would have been $V_n = V_0 - n \times M$. Within each scenario, we compute V_n for various values of n and various levels of communication. This will allow us to address

73

several important questions, one of which being: for each value n of communication cost, what is the optimal level of communication?

As to the issue of differences in objectives of players, one way to model this would be to make each player responsible for a subset of the enemy targets in the sense that his total score will be equal to the value derived from only those targets. The players may have several (or more) targets in common, as seen below:

Targets in Common

T_1 T_4 T_6
T_5

Sphere of influence ───→   ←─── Sphere of influence
of Player 1   of Player 2

T_2 T_7
T_3 T_8

L_1 L_2
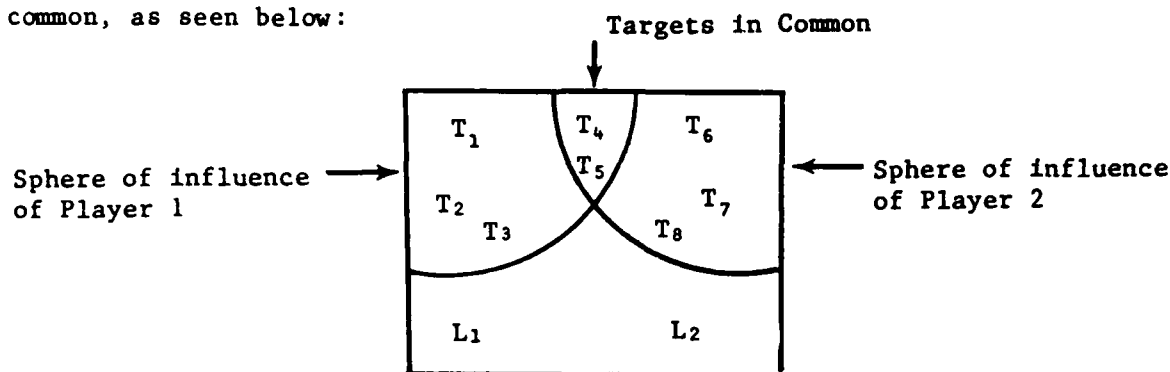
As far as the information is concerned, each player could be allowed only partial information about the other player's resources, or he could be allowed complete information. Similarly, the cost of communication could be significant, or virtually zero. These parameters will be important, but not crucial. Even with perfect information and communication, the problem of coordination between several players with partially conflicting objectives will be apparent in all its glory, and we will investigate how the players will negotiate in order to help each other (by attacking common defenses, by negotiating their attacks on the common targets, etc). Investigating this setup will help us not only understand the problem of conflicting goals, but also the larger problem of assigning responsibility within a hierarchy. For example, consider the case of one commander with two subordinates:

1

2   3
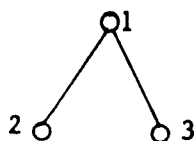
The objective of the commander is to maximize the value of the attack mission. His main role will be delegating the responsibility, i.e., to divide the tasks of the attack mission between his two subordinates and to monitor

74

their execution. This can be done by making each of the subordinate's utility function depend only on his subset of "targets of responsibility" like in Figure 2, and by allowing the subordinates to coordinate their efforts.

A good commander is thus the one who is good at decomposing the problem, at assigning the right utility functions to each subordinate, and at being able to adjust these utility functions in cases when the subordinates are at an impasse, or are not doing the right things all together.

To give a small example of some issues involved, let us consider the following question: with respect to the common targets, should each player be given the number of points equal to the total number of points extracted from these targets by both players, or equal to his fair share of (his contribution to) these points? The first approach may result in both players over-emphasizing the importance of the common targets or in Player A's trying to distract or prevent Player B from attacking these targets, since this will automatically increase Player A's score.

These, and other issues, will be investigated when the distributed interactive decision making system is fully implemented.

5. AUTOMATED DECISION MAKING TECHNIQUES

This section discusses several techniques for automating the DDM process. Most of the techniques to be discussed have foundations in mathematics, the other two originated from artificial intelligence.

In order to completely automate decision making for the problem described in Section 4, some simplifying assumptions were made. The one assumption that applies throughout this section is that the launch platforms are stationary. Further development may allow us to remove this assumption. Any other assumptions required by specific automated decision making techniques will be mentioned as necessary.

5.1 SEQUENTIAL ASSIGNMENT

5.1.1 Description

The sequential assignment algorithm chooses a set of missile assignments, one at a time, in a greedy but myopic fashion. It is greedy in that during each assignment cycle, it chooses that assignment yielding the highest expected return; it is myopic in that it does not consider the effect that the current decision could have on future assignments.

Initially, only targets are considered as possible objectives since defenses have no explicit value. During the first assignment cycle, the expected return associated with each possible assignment of a missile to a target is calculated based on the probability of surviving the optimal path to the target, the probability of then destroying the target, and the value remaining to the target. The assignment of missile to target having the largest expected return is determined, with ties being broken arbitrarily. The chosen assignment, say missile M1 to target T1, is irrevocably made, though not executed immediately. The value remaining to T1 for later calculations is reduced appropriately.

In all subsequent assignment cycles only those missiles without assignments are considered. Potential assignments of missiles to targets are evaluated as before. In addition, defenses are now treated as allowable objectives. Determining the expected value derived from attacking a defense is more difficult than for a target. Consider assigning missile M2 to defense

D1 in the second assignment cycle, remembering that the assignment of M1 to T1 has not actually been executed yet. We calculate the probability of existence for D1 after a hypothetical attack by M2 and calculate the expected return for the assignment M1 to T1 given that D1 has been probabilistically weakened. The net improvement in the assignment of M1 to T1 is the value given to the assignment of M2 to D1. The assignment of missile to target or defense with the largest expected return is the one chosen in the second assignment cycle.

In general, the value associated with attacking a defense is calculated by determining the net improvement to all previously chosen assignments of missiles to targets. Note that the assumed order in which missions are to be executed is not the order in which they are chosen. The evaluation procedure assumes that all missions against defenses are executed first, in the order in which they are chosen, followed by attacks on targets in any order. During any cycle, if the chosen assignment is to a target, only the value remaining to that target need be updated; if the assignment is to a defense, the probability of existence of that defense, all optimal paths affected by attacking that defense, and the value remaining to any target with a previously assigned missile that is affected by attacking that defense must be updated.

After running through as many assignment cycles as there are missiles, the algorithm terminates. Because the algorithm is myopic, there is no guarantee that the solution that it arrives at will be optimal. In its current implementation, it is best suited for use by a decision maker given the local problem of assigning a number of missiles to objectives, then varying the solution to coordinate with other decision makers. In principle, the algorithm could also be implemented by allowing each missile to have a microprocessor that determines the assignment for that missile. To calculate the effect that assigning a missile to a defense would have on other missiles that already had assignments to targets would require a large amount of communications. A practical implementation of this form would most likely require that we develop some means for approximating the effect of attacking a defense that would reduce the communications needs.

5.1.2 Example

We now illustrate the sequential assignment algorithm with an example. Figure 5-1 contains the same board configuration used to illustrate our interactive planning system in Section 4. For this algorithm, the launchers are assumed fixed and timing issues are not explicitly considered, although timing could be determined after the set of missile assignments was chosen.

Initially, only targets are considered as objectives, with target T2 being the target yielding the largest value, as seen in Figure 5-2. Missile M5 is therefore assigned to T2.

Note that in the second assignment round, as shown in Figure 5-3, defenses are now allowable objectives. Attacking D1 or D6 would not help M5 reach T2, but attacking any other defense would increase the expected performance of M5 by the amount shown. Higher value can be derived from T3 though, so the second assignment sends M3 against T3.

The process continues until each missile has received an assignment. The final solution sees two missiles attacking D4, one attacking D2, and the remaining five missiles attacking targets for a total expected return of 441.6 (Figure 5-4).

5.2 SEQUENTIAL REASSIGNMENT

5.2.1 Description

This is a modification of the sequential assignment algorithm. It is guaranteed to find a solution at least as good as that found by sequential assignment; in many cases it will find a better solution at the cost of some additional computations.

In sequential assignment, only missiles that have yet to receive assignments are considered in each assignment cycle. The assignment in each cycle is chosen based on those assignments already made in previous cycles. It may be the case that the objective chosen for a given missile might not have been chosen had it been known at the time that some other assignment was going to be made during a subsequent assignment cycle. For example, the missile that was assigned to some objective in the first cycle based on the initial situation may be able to derive greater value from a different objective given knowledge that an assignment was made to some defensive site

78

```
                                          1  T1  100    0      T2  200    0
     T2                      T4           2  T3  150    0      T4  100    0
 T1     D2           T3                   3  D1  .99           D2  .99
              D3          D5              4  D3  .99           D4  .99
 D1              D4           D6          5  D5  .99           D6  .99
                                          6  L1  .99  Y   M1 M2
                                          7  L2  .99  Y   M3 M4
 L1              L3                       8  L3  .99  Y   M5 M6
                      L4                  9  L4  .99  Y   M7 M8
       L2         .                      10
                                         11
                                         12
                                         13
 1  2  3  4  5  6  7  8  9  10 11 12 13
```
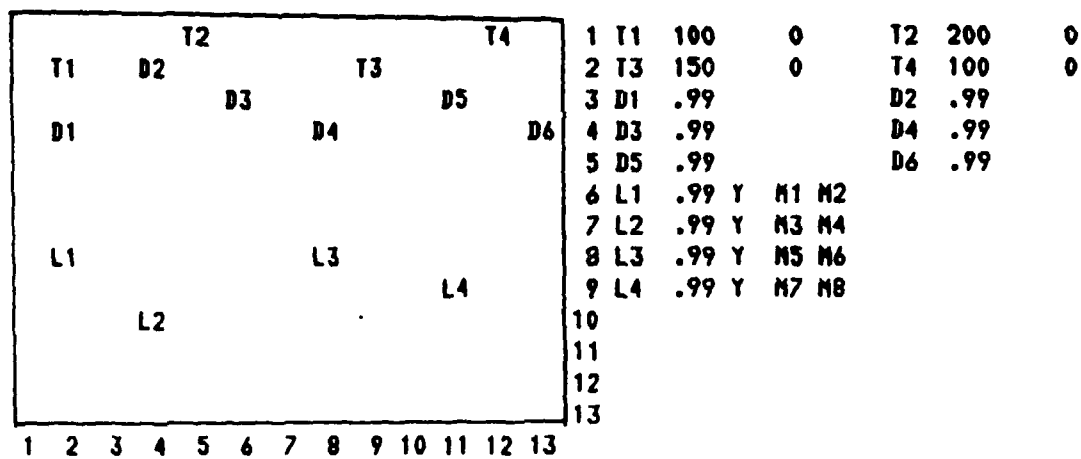
Initial fuel is 25

Figure 5-1   Board Configuration for Example

```
target      1 score  51.2 points.  Current target value 100.0
target      2 score  81.9 points.  Current target value 200.0
target      3 score  76.8 points.  Current target value 150.0
target      4 score  51.2 points.  Current target value 100.0


missile     5 to target      2, value  81.9 points
```

Figure 5-2   First Assignment Cycle


```
target      1 score  51.2 points.  Current target value 100.0
target      2 score  48.4 points.  Current target value 118.1
target      3 score  76.8 points.  Current target value 150.0
target      4 score  51.2 points.  Current target value 100.0
defense     1 score    .0 points
defense     2 score  33.6 points
defense     3 score  10.5 points
defense     4 score  24.5 points
defense     5 score  24.5 points
defense     6 score    .0 points


missile     3 to target      3, value  76.8 points
```

Figure 5-3   Second Assignment Cycle


```
Assignment Summary
  missile     5 to target      2 value  81.9 points.
  missile     3 to target      3 value  76.8 points.
  missile     1 to target      1 value  51.2 points.
  missile     7 to target      4 value  51.2 points.
  missile     2 to defense     4 value  62.9 points.
  missile     6 to target      2 value  49.8 points.
  missile     4 to defense     4 value  40.8 points.
  missile     8 to defense     2 value  27.1 points.


      Total Score      441.6
```

Figure 5-4   Final Assignments

80

during the fifth assignment cycle.

In sequential reassignment, maximum expected returns are calculated for both previously assigned and previously unassigned missiles during each assignment cycle. Previously assigned missiles are considered first. If at least one of these can achieve expected net improvement by changing its assignment, then the missile with the greatest possible net improvement is reassigned during that cycle, values are updated accordingly, ard the next assignment cycle begins. If no improvement is possible through reassignment, then the best previously unassigned missile is chosen, as in sequential assignment.

This algorithm must converge, though again not necessarily to the optimal solution. However, it does allow some mistakes in missile to objective assignments to be corrected, as illustrated in the example that follows.

5.2.2 Example

The advantages of sequential reassignment can be seen clearly in the example contained in Figure 5-5. In this example, M1 has enough fuel to reach T1 or T2, but M2 can only reach T2. The first assignment is M1 to T2, since T2 is the higher valued target. No choice remains but to assign M2 to T2 also, yielding only an additional 32 points. At this point, it becomes clear that M1 would be better utilized against T1, because M2 can do an adequate job against T2 without the help of M1. By reassigning M1, an additional 108 points can be derived for a total of 300 points. Note that without reassignment, the score would only be 192 points.

5.3 NEGOTIATION ALGORITHMS

5.3.1 Description

Negotiation algorithms encompass a variety of related algorithms. In each, it is assumed that there are N decision makers. Each decision maker is given control of some subset of the total resources; each decision maker also has a subset of the objectives, targets and defenses, to which missiles can be assigned. Assume that each decision maker has solved its own local subproblem by some means, for instance by applying sequential assignment or reassignment. Negotiations for resources between neighboring decision makers commence at

81

```
 ┌──────────────┐
 │ T1     T2    │1
 │              │2          T1  175      0        T2  200      0
 │              │3          L1  .99 Y  M1
 │         L1   │4          L2  .99 Y  M2
 │          L2  │5
 └──────────────┘
  1  2  3  4  5
```

Initial fuel is 10

target   1 score 140.0 points. Current target value 175.0
target   2 score 160.0 points. Current target value 200.0

missile   1 to target   2, value 160.0 points

target   1 score -20.0 points. Current target value 175.0
target   2 score 32.0 points. Current target value 40.0

missile   2 to target   2, value 32.0 points

target   1 score 108.0 points. Current target value 175.0

missile   1 to target   1, value 108.0 points

Assignment Summary
missile   1 to target   1 value 140.0 points.
missile   2 to target   2 value 160.0 points.

     Total Score   300.0

Figure 5-5   Sequential Reassignment Example

82
```

this point.

Several means exist for implementing the negotiations, and the various
negotiation algorithms vary in this respect. Perhaps the simplest method is to
consider the decision makers one at a time. Decision maker 1 chooses a
missile to give up -- the choice can be arbitrary, or it could be the missile
contributing least to decision maker 1's performance, or the last missile
assigned by the sequential assignment algorithm, etc. Decision maker 1
determines the degradation suffered if the chosen missile were to be given up.
The net loss can be thought of as a "selling price." The missile is then
offered to each of decision maker 1's neighbors. Each neighbor calculates its
"buying price" by calculating the net gain induced by having the additional
missile to allocate in its local subproblem. Each neighbor transmits its
buying price to decision maker 1. If the maximum buying price exceeds the
selling price, then overall performance is improved by transferring control of
the missile from decision maker 1 to the highest bidder. If no neighbor offers
more than the selling price, then no change occurs. This process is repeated
once for each decision maker, then returns to decision maker 1 for another
cycle. If no missiles are transferred during a negotiation cycle, the
algorithm terminates.

More complicated means for choosing the missile to be relinquished exist.
For instance, decision maker 1 could offer all of its missiles "for sale."
Each neighbor could submit a list of bids, one for each missile. The missile
transfer yielding the largest net gain would then be effected.

We are uncertain as to which implementation of missile negotiation is the
best. Different versions can be evaluated with respect to the overall
performance of the final solution generated, the number of calculations
required, and the number of communications required. These issues may be
investigated at a later time.

An example of one implementation of missile negotiation follows.

5.3.2  Example

Consider again the board configuration used in the examples in Section 4
and in Section 5.1. Suppose that resources and objectives are distributed

among four decision makers as shown in Figure 5-6. For the example we use the
algorithm in which each decision maker uses sequential assignment to solve its
local subproblem and it is the last missile to receive an assignment that is
offered to other decision makers. We assume a complete network of decision
makers, i.e., any two decision makers are neighbors.

Figure 5-7 contains the local solution found by each decision maker
before negotiations begin. Decision maker 1 begins the negotiation by
offering missile M2 at a price of 25. Offers of 47, 19 and 16 are received
from decision makers 2-4, respectively. Missile M2 is therefore given to
decision maker 2, where it is assigned to T2 while M4 now becomes assigned to
D2. (Missile 4 gets this new assignment because sequential assignment is run
again for decision maker 2 with the additional resource M2.) Each decision
maker repeats the same process, but no more missile transfers occur beyond
this point and the algorithm terminates as in Figure 5-8, with negotiation
yielding an improvement of 21.6 points.

The reason for transferring control of M2 to decision maker 2 is
intuitively obvious. Decision maker 1 has responsibility for T1, valued at
100 points; decision maker 2 has T2 worth 200 points. However, each was
initially allocated two missiles. Therefore, decision maker 2 had a greater
need for resource than decision maker 1, and received additional resource
through negotiation. On a larger example, one would expect many missile
transfers to occur before termination of the algorithm.

## 5.4  MARGINAL UTILITY

Marginal utility algorithms are commonly used for resource allocation
problems. Suppose that a number of decision makers are competing for the use
of global resources and that the overall objective function is separable.
Suppose further that for any given resource allocation, each decision maker
can determine its own marginal utility (resource price), i.e., the price it
would pay or would accept in exchange for an incremental unit of resource.

Assume that the utility function for each decision maker is concave.
This means that the lower the resource allocation (supply), the higher the
marginal utility (price). Then marginal utility theory states that a
necessary and sufficient condition for a set of resource allocations to be

Initial fuel is 25

Figure 5-6   Board Configuration for Example

```
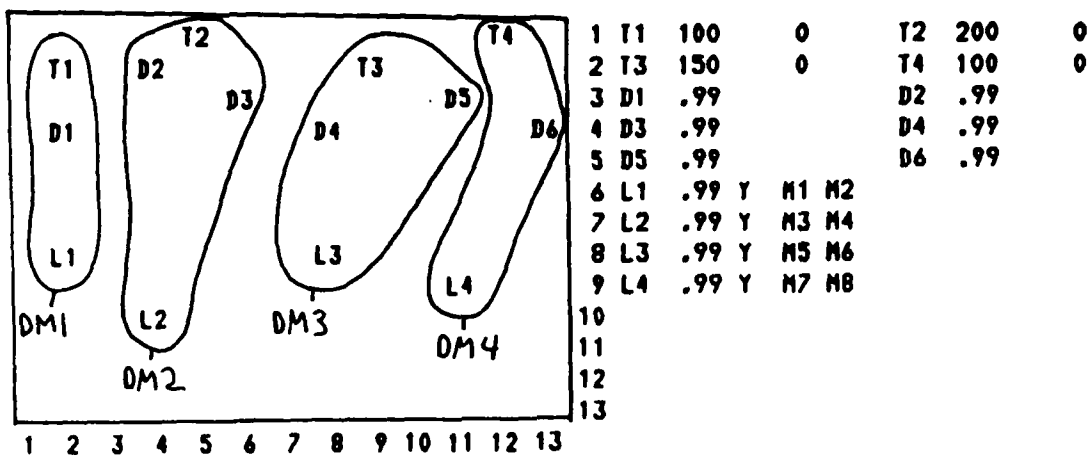            Analysis for Decision Maker Number,  1
missile     1 TO target     1 VALUE  51.2 points.
missile     2 TO target     1 VALUE  25.0 points.

     Total Score     76.2




            Analysis for Decision Maker Number,     2
missile     3 TO target     2 VALUE  71.7 points.
missile     4 TO target     2 VALUE  46.0 points.

     Total Score     117.7




            Analysis for Decision Maker Number,     3
missile     5 TO target     3 VALUE  76.8 points.
missile     6 TO target     3 VALUE  37.5 points.

     Total Score     114.3




            Analysis for Decision Maker Number,     4
missile     7 TO target     4 VALUE  51.2 points.
missile     8 TO target     4 VALUE  25.0 points.

     Total Score     76.2



            Total score     384.4
```

Figure 5-7   Pre-negotiation Assignments

```
              Analysis for Decision Maker Number,  1
Missile      1 TO target      1 VALUE  51.2 points.

    Total Score    51.2




              Analysis for Decision Maker Number,    2
Missile      3 TO target      2 VALUE  71.7 points.
Missile      2 TO target      2 VALUE  46.0 points.
Missile      4 TO defense     2 VALUE  46.6 points.

    Total Score    164.3




              Analysis for Decision Maker Number,    3
Missile      5 TO target      3 VALUE  76.8 points.
Missile      6 TO target      3 VALUE  37.5 points.

    Total Score    114.3




              Analysis for Decision Maker Number,    4
Missile      7 TO target      4 VALUE  51.2 points.
Missile      8 TO target      4 VALUE  25.0 points.

    Total Score    76.2



            Total score    406.0
```

Figure 5-8 Post-negotiation Assignments

END

FILMED

DTIC

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

optimal is that each decision maker has the same marginal utility, this value being referred to as the "equilibrium price." Marginal utility algorithms perturb the resource allocations systematically until the equilibrium price is achieved.

However, the missile assignment problem does not satisfy the conditions required by conventional marginal utility algorithms. The objective function is not separable: a decision maker's return may depend on the decisions made by other decision makers. Also, the resource to be allocated (missiles) is discrete rather than continuous. This means that marginal utility cannot be defined incrementally as the derivative of the return function and that a decision maker's buying price and selling price for resource may differ. A version of marginal utility more suited to the missile assignment problem is described in the appendix.

5.5 BRANCH-AND-BOUND SEARCH TECHNIQUE

This is a tree search technique whose origins lie in artificial intelligence and in optimization theory. It eliminates from consideration entire sub-trees by using a technique called branch-and bound. Here is a brief description of the algorithm.

The algorithm construct... tree of possible assignment decisions. Each node of the tree corresponds to some subset of assignments. An arc in the tree correspond to one new assignment. As the tree is constructed, we calculate upper and lower bounds on the total expected performance originating from any given node. If the upper bound on performance for any node i is lower than the lower bound for some other node j, then we can drop node i from the tree, thus pruning out all branches that would emanate from node i.

There are various ways of computing lower and upper bounds on performance, since these bounds are not unique. In fact, the success of the branch-and- bound method depends on the tightness of these bounds, i.e., on how closely these bounds approximate the real value for optimal performance. Methods of Artificial Intelligence may provide useful tools for finding tight upper and lower bounds.

88

Analysis for Decision Maker Number, 1

Missile   1 TO target   1 VALUE 51.2 points.

    Total Score   51.2

Analysis for Decision Maker Number,   2

Missile   3 TO target   2 VALUE 71.7 points.
Missile   2 TO target   2 VALUE 46.0 points.
Missile   4 TO defense   2 VALUE 46.6 points.

    Total Score   164.3

Analysis for Decision Maker Number,   3

Missile   5 TO target   3 VALUE 76.8 points.
Missile   6 TO target   3 VALUE 37.5 points.

    Total Score   114.3

Analysis for Decision Maker Number,   4

Missile   7 TO target   4 VALUE 51.2 points.
Missile   8 TO target   4 VALUE 25.0 points.

    Total Score   76.2

Total score   406.0

Figure 5-8   Post-negotiation Assignments

5.6 AI MISSION PLANNING SYSTEM

5.6.1 Expert Systems

The algorithms described above have a strong analytical flavor in the sense that, even though they are not guaranteed to be optimal, their structures are fairly straightforward and their strengths lie in the complex mathematical computations utilized by them. This approach contrasts with the artificial intelligence approach which relies more on higher-level conceptual rules of inference and planning, similar to the way the human mind works, rather than on intense mathematical computations.

The most promising AI approach involves building expert rule-based systems. These systems are created by having experts (in our case, military decision makers) share the rules they use in their decision making. These rules are then put together by computer scientists into a coherent rule-and data-driven computer algorithm. As the name indicates, this algorithm has a very complex (unpredictable in advance) flow of control, depending on the data and the rules that were applied to that data. The result is a highly flexible, "intelligent" decision system. Two words of caution are necessary. First, unfortunately all heuristic rules have many exceptions. Thus, the expert system is bound to make bad decisions some of the time, just as human decision makers do. Second, although fallible, the human decision making process is incredible complicated and still very much a mystery to science. At the present it is impossible to incorporate all, or even most, of the human intelligence involved in mission planning (or in any other intelligent endeavor for that matter).

Therefore, success of a particular expert system depends on the degree to which the most important decision making concepts have been captured, together with their interrelations. That requires a major research effort, good human experts to share their experience, and a lot of attention to detail.

In the next subsection we are going to describe high-level blueprints for a particular expert-based mission planning system. The system was created on the basis of some of the experience we've derived from playing the DDM game. As we gain more planning experience and as more military experts share their experience with us, the expert system is bound to become much more detailed

and complicated, and even its high-level structure may undergo dramatic changes. Thus, what follows should be considered as high-level example of an expert system.

5.6.2 Design of an AI Expert System for DDM Mission Planning

This system consists of seven experts working as a team in solving the posed mission planning problem. Here is a description of the roles played by these experts.

1) Topological connectivity evaluator (or pocket dividor): Topologically divides the enemy targets into pockets such that targets within each pocket are reachable from one another without going through major defensive threats.

2) Pocket Specialists: Each pocket is dynamically assigned a pocket specialist that comes up with the marginal missile utility curve, i.e., determines the return $R(N)$ extractable from the pocket given N units of resource (i.e., missiles) and assuming that these missiles units got safely inside the pocket.

3) Ring-Cutting Experts: There is one ring-cutting specialist per pocket. This expert evaluates various paths of getting inside the pocket (cutting through the ring), and chooses several (by N-best technique for thresholding, or clustering) of the most promising paths.

4) Cooperative Planning Expert: This expert receives the sets of best paths from each ring-cutting expert and combines them into cooperative ring-cutting plans. It chooses several of the best of these plans on the basis of their minimality and cooperation, i.e., needing to destroy as few defenses as possible to accomplish the cutting, and also on the basis of the importance of the pockets that will be accessible through these cuts.

5) Mission Planning Expert: Evaluates each of the cooperative ring-cutting schemes passed on by Expert 4 and optimizes the missile assignments (i.e., how many defenses to attack, which of the targets, etc.) within each scheme using the marginal pocket utility curves passed on by pocket evaluators, and tries to choose the most valuable overall assignment.

6) Constraint Expert: For the proposed assignments, checks if all the constraints (like fuel constraint, time constraint, launcher constraint, weapons constraint, etc.) are satisfied. If this best plan is satisfiable, then we're done. If not, it notes which constraint is not satisfied, imposes it on Expert 5 and tells Expert 5 to re-do his mission planning with the added constraint. Expert 5 may also call the pocket and ring-cutting planners for re-evaluation, if necessary. Thus, we'll have iteration to a solution.

7) Meta-level Planner: Coordinates the performance of the above experts. Calls them in the right sequence and in parallel (like experts of Type 2 and 3). If Expert 6 comes up with new constraints, the Meta-planner will call Expert 5 and the Pocket Experts and Ring-cutters for re-evaluation with the added constraints.

This present structure represents a transition between mathematical algorithms and flexible AI systems. It already has a fairly flexible inter-expert control structure, but this structure will get much more flexible as the system becomes more detailed. It also has a mixture of mathematical and rule-based experts, with experts number 1 and 2 being fairly mathematical, and experts 4 and 6 being strongly rule-based. We should also notice that this expert system has a strong distributed flavor. As we saw, there are several pocket and ring- cutting experts working in parallel with each other, one of each expert for every pocket. These experts come up with local attack plans, and these plans are then merged and reconciled by the mission planning expert and the constraint expert. This observation points out a way for a natural implementation of this expert system in the distributed decision making form.

6. DESIGN METHODOLOGY

6.1 INTRODUCTION

In this section we present a methodology for designing distributed decision making systems. Given a scenario where decision making is involved, it is desired to find the appropriate decision making structure. This includes the specification of the task structure, authority structure, information structure and communication structure. A baseline design methodology was suggested in the original proposal for this project. This section summarizes the improvements we have made on the design methodology since then.

The structure of this section is as follows. In Section 6.2, we discuss the general design problem and the design process. The key steps in the design process are identified, together with aids which would be useful for the various steps. Section 6.3 describes the design problem for distributed decision making. This problem has unique features which impact upon the design process. The specialized design process is then described in detail. Section 6.4 contains an example to illustrate how design is accomplished.

6.2 GENERAL DESIGN PROBLEM

Design is a process of synthesizing new forms from existing ones in response to a certain function or goal, while satisfying other constraints. It is the dominant activity in engineering as well as other professional di _,lines such as architecture, and distinguishes these disciplines from the natural sciences. Our interest in design is primarily on distributed decision making systems. However, a discussion of the general design problem will enable us to understand better the specific design process needed for distributed decision making.

6.2.1 Design Problem

A design problem involves the following main ingredients: an object (system) to be designed, the environment in which the object (system) functions, and the requirements on the design. To be more precise, let S be the system or object to be designed, and E be the environment under consideration. The starting point of the design problem is given by the

92

triple (S,E,F) where F is the set of requirements defined on S and E. As an
example, S can be a radio, E be the relevant environment and F be the
performance requirements for S in the environment E. The output of the design
problem is a design, s, which when implemented, would fulfill the requirements
F. The design s would have associated specifications f defined on the
environment e. Stated another way, the design problem is: Given (S,E,F),
find a realization (s,e,f) such that with aggregation, (S,E,F) is achieved.
(s,e,f) provides all the necessary specification for the construction of the
system. Usually, in a design problem, one fixes the level of the design by
constraining e. Depending on the level of description of e, we can have a
high or low level design.

Almost all design problems, whether they are engineering systems,
architectural designs, construction of computer programs, or even composition
of music, can be represented in this form. Figure 6-1 represents the design
problem. Note that we have viewed design as a "satisficing" problem, rather
than optimization, since in all but the simplest cases, optimization is
neither necessary or possible.

Figure 6-1   Design Problem

93

6.2.2 Design Process

The design process transforms the original system (S,E,F) to a final design (s,e,f), which satisfies the original requirements. Usually this is carried out in a number of levels, starting from the original problem description, going through increasingly more detailed designs until a final design is obtained. The number of levels depends on the complexity of the problem. When the original problem is quite complex and a detailed design is needed, many intermediate levels may be needed as in Figure 6-2.

$$
\boxed{(S,E,F)}
$$

$$
\downarrow
$$

$$
\boxed{(S_1, E_1, F_1)}
$$

$$
\downarrow
$$

$$
\boxed{(S_2, E_2, F_2)}
$$

$$
\vdots
$$

$$
\boxed{(s,e,f)}
$$

Figure 6-2   Multilevel Design

The output of each level serves as the input to the next level. Usually, one has a pretty good idea about the number of levels needed and the kind of description at each level. For example, in a lot of engineering systems, one may go through the behavioral, functional and physical levels. In the following we shall focus on the process of going from one level to the next.

We assume a design (S_i, E_i, F_i) is already given and that the next level design $(S_{i+1}, E_{i+1}, F_{i+1})$ which is consistent with (S_i, E_i, F_i) has to be found. Four steps are usually needed for this (see Figure 6-3). They are: synthesis, optimization, evaluation and selection. These steps will now be described in detail.

(S_i, E_i, F_i) high level design

```
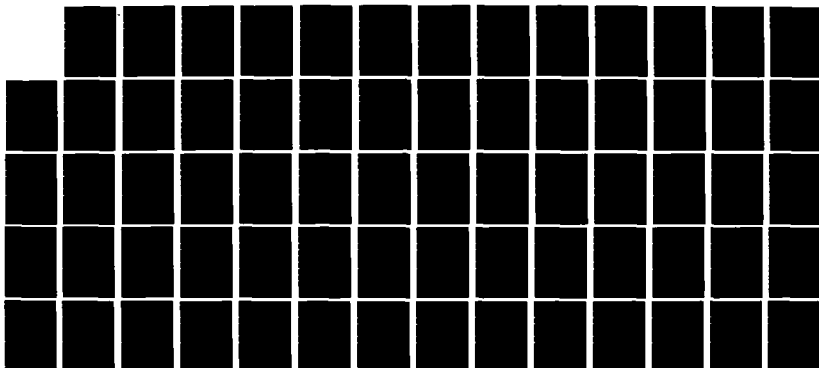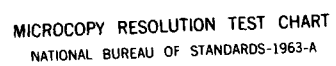                    ┌─────────────────────┐
               ┌───→│      synthesis      │
               │    └─────────────────────┘
               │               │
               │               ↓
               │    ┌─────────────────────┐
               │    │     optimization    │
               │    └─────────────────────┘
               │               │
               │               ↓
               │    ┌─────────────────────┐
               └────│      evaluation     │
                    └─────────────────────┘
                               │
                               ↓
                    ┌─────────────────────┐
                    │      selection      │
                    └─────────────────────┘
                               │
                               ↓
```

$(S_{i+1}, E_{i+1}, F_{i+1})$ low level design

Figure 6-3   Design Process at Each Level

6.2.2.1 Design Synthesis

In this step, possible candidate designs are generated starting from a higher level design. The candidate designs should include one which would eventually satisfy all the specifications. Furthermore, to reduce the tasks of optimization, evaluation and selection, the set should be reasonably small. Synthesis is a creative process except in very trivial problems. It is also the most important step since a good candidate can cut down the design time tremendously. In synthesis, one almost needs the knowledge to look ahead to generate a design which is likely to succeed. This requires a very good understanding of the problem as well as experience in design. Analytical tools are not particularly useful in this step. Aids, however, can be provided to assist in the cognitive process of synthesis. The following are some possibilities.

a. Representation of the problem.

One may view the design process as a transformation from one (usually more abstract) representation of the system into another (usually more detailed). If the original representation is presented in a proper way, the next-level representation can be cast into a mathematical framework, and then mathematical techniques can be used to reveal the next level representation. This is, for example, the approach suggested in [152], where a decomposition technique is used to partition the requirements of a design into almost independent subsets which can then be satisfied individually. In any case, careful analysis and clever representation can assist the designer tremendously.

b. Design Rules

Although there is no hard scientific way of generating the next level design, in many application areas, a large body of knowledge has been accumulated through the solution of design problems. This knowledge is usually available as design rules of thumb in the respective areas. A design can benefit from the use of these rules. In fact, an experienced designer is an individual who has a large number of rules to help him in generating new designs. Conceivably, an expert artificial intelligence system can be built to imitate the performance of an expert designer.

c. Focusing of Attention

As seen in Figure 6-3, design is seldom a one-shot process. After the design alternatives have been evaluated and rejected to be unsatisfactory, a new design alternative has to be generated. The reasons for rejecting the old designs are extremely useful for generating the new design. The new design can be synthesized by removing the unsatisfactory parts in the old design. Instead of finding a design which can satisfy all the requirements, one looks at a smaller problem of modifying a design to remove certain "misfit" variables. This is closely tied to the satisficing nature of the design process.

6.2.2.2 Design Optimization

The result of the synthesis is a design alterna e $(S_{i+1}, E_{i+1}, F_{i+1})$. Frequently, some of the design parameters still rema   o be specified. This can be chosen by optimizing a subset of the requirem F_i with respect to the parameters. The optimization part can be formal   quite easily. The parameters belong to S_{i+1}. The environment as specified by E_{i+1} may be completely deterministic or it may contain some uncertain parameters. The objective function for the optimization may be one measure which the designer considers to be important for the performance of the system. Rigorous mathematical techniques are available for optimization. For deterministic problems, mathematical programming can be used. With uncertainty which is probabilistic in nature, statistical decision theory is applicable. For problems which are not very well structured mathematically, one may want to use heuristic search methods. Overall, optimization is probably one of the easiest processes to automate. Aids can be provided in both mathematical optimization methods and heuristic search.

6.2.2.3 Design Evaluation

After a design $(S_{i+1}, E_{i+1}, F_{i+1})$ has been optimized, one can evaluate its performance in terms of the higher level requirements F_i. At the highest level, these requirements may include functional correctness, cost, performance, etc. At other levels, they will be quantities which are appropriate for the degree of description required. The design is evaluated with respect to the requirements F_i. If all the requirements are satisfied,

the design alternative is admissable. Otherwise, the designer would have to generate a new design and repeat the process of synthesis, optimization and evaluation all over again.

In evaluation, the design is completely specified subject only to the uncertainty at level i+1. Many tools are available to facilitate evaluation. The following are some typical ones.

a. Simulation

If the design is complicated, evaluation cannot be performed analytically. Simulation is a powerful tool to be used in these situations. Monte-Carlo simulations are frequently used when uncertainty is present in the environment. Note that even though the same set of attributes is available during the optimization step, for practical reasons, only a smaller set is usually considered. During evaluation, however, the complete set should be used. For the same reasons, sometimes simpler models are used for optimization while the complete models are used in evaluation.

b. Experimentation Facilities

In certain situations, it may be desirable to build a testbed to evaluate a design. This is the case when human beings (as part of the environment) are involved in using the system. Models can be used to replace the human beings, but a test bed with human participation will be more realistic.

6.2.2.4 Design Selection

If more than one design alternatives are available after the evaluation step, it is necessary to select a design to be implemented (when one has already reached the lowest level) or to be further expanded into a lower level design (when one is still at an intermediate level). A design which survives the evaluation step satisfies all the design requirements, and is thus acceptable. To select one design from many acceptable designs, one has to look at the requirements themselves and how they have been satisfied by the various designs. Consider the following example. Suppose two acceptable designs are (s_1, e_1, f_1) and (s_2, e_2, f_2). Suppose the requirements F in the original problem are represented by the vector $F=(F^1, F^2)$. Let $F^1 (s_j, e_j, f_j)$ represent the evaluation of design j with respect to the requirement F^1,

98

and $F^1(s_1,e_1,f_1) > F^1(s_2,e_2,f_2)$ if design 1 is better than design 2 with respect to requirement i. Suppose

$$F^1(s_1,e_1,f_1) > F^1(s_2,e_2,f_2)$$

$$F^2(s_1,e_1,f_1) < F^2(s_2,e_2,f_2)$$

We have to decide on one particular design. The selection or choice process depends on which requirement we consider to be more important. If F^1 is more important than F^2, then design 1 should be chosen, and vice versa.

In general, the selection step involves the introduction of a higher level criterion than is represented by the requirements F. Some useful tools to assist in the selection are:

a. Sensitivity Analysis

Here the higher level criterion is the sensitivity of the design to certain variable unknown environmental parameters. In general, a design whose performance is too sensitive to assumed parameter values is undesirable.

b. Trade-off Studies

One would like to see how the performance attributes change as one goes from one design to the other. The standard cost-benefit analysis falls into this category.

c. Multi-attribute Utility Theory [33]

Here one constructs a global utility function starting from the more elementary performance attributes. This utility function is then used as the high level criterion to select the right design.

d. Analytic Hierarchy Process [153]

In this case, a higher level criterion (goal) is specified. Instead of constructing a multi-attribute utility function, one assigns weights to the overall contribution of the various attributes (requirements) to the higher level goal or criterion. This is accomplished by first forming a hierarchy,

99

with the goal of the system at the highest level and the attributes F at the lowest level. By con. dering pairwise comparisons of two factors at each level with respect to a goal at a higher level, one can arrive at their relative weights. With these weights, comparison between designs is then possible.

So far we have considered the activities at one level. In a multi-level design process, these four steps are repeated at each level. Frequently, given a design (S_i, E_i, F_i), one may not be able to find a lower level design $(S_{i+1}, E_{i+1}, F_{i+1})$ which will meet the requirements. In this case, it is necessary to find a different design at level i. By considering the requirements in F^i which cannot be satisfied, one can find a design which has a better chance of being satisfactorily realized.

In a multi-level situation, another high level criterion to be used for selection of a design can be the promise of the current search path, i.e., how likely that the selected design can lead to an acceptable final design. In order to avoid a lot of backtracking, it may also be desirable to select more than one design for future expansion at each level. Figure 6-4 illustrates a depth first type multi-level design process.

6.3 DISTRIBUTED DECISION MAKING SYSTEM

In this section we apply the general design methodology described in Section 6.2 to the design of distributed decision making systems. Our presentation follows the structure of Section 6.2. First, the design problem is described. This is then followed by a discussion of the design process.

6.3.1 Design Problem

Two features distinguish this design problem from a general problem. First, the system is used for decision making. Second, the system is distributed in nature. This will be used in the problem definition.

The general scenario that we are interested in is shown in Figure 6-5. Sensors generate measurements from the external world. Effectors are the means by which the external world is affected. The decision makers may be in place already, or they may be part of the design. The objective is to design a distributed decision making system. Communication links will be provided to

100

PROBLEM DESCRIPTION
(S,E,F)

(S_1,E_1,F_1)

(S_2,E_2,F_2)

(S_3,E_3,F_3)

(s,e,f)

FINAL DESIGN

Figure 6-4   Example of a Depth First Search

101

DM DECISION MAKER

S SENSOR

A EFFECTOR

Figure 6-5 Environment of Decision System

connect sensors, effectors and the decision makers in some fashion. Computers will assist in the decision making. This is illustrated in Figure 6-6.

In the terminology of Section 6.2.1, S is the distributed decision making system to be designed. E, the environment, includes the sensors, effectors, as well as the decision makers if they are already in place. Obviously, it also includes the external world in which the system operates. F, the attributes and requirements, include the goals of the system as well as other constraints imposed by E on S.

In terms of a tactical air battle scenario, the external world consists of the enemy targets and their defenses. The sensors are the radars and other sensing devices, while the effectors are the planes or missiles (resources which affect the external world). The goals of the system can be the destruction of the enemy targets and the constraints may be related to fuel consumption, geographical separation, etc.

The desired output of the design problem is the triple (s,e,f) which is consistent with (S,E,F); s specifies all the components in the distributed decision making system, e is the environment in which they operate and f gives the requirements on the components and the environment. Specifically, this description will include:

- Task architecture: What is each decision maker responsible for?

- Authority architecture: What resources does each decision maker control and does a decision maker control other decision makers?

- Information structure: What does each decision maker know?

- Communication structure: How do the decision makers communicate with each other?

In addition, we may also have specifications on the other components (such as computers) in the system. Usually the design can stop here. However, if so desired, we can also continue to design algorithms used by the decision makers and/or computers.

Figure 6-6 Distributed Decision Making Architecture

Note that although the problem is one of designing a distributed decision making system, our discussion here is limited to a centralized design methodology. Certain parts of the design methodology can be distributed, but this will not be elaborated here.

6.3.2 Design Process

Distributed decision making systems usually arise in complex scenarios. Thus, one would expect a multi-level approach to be needed for the design process. The number of levels depends on the particular problem under consideration. However, for many situations, a three level process will serve very well as the starting point. The three levels are shown in Figure 6-7.

Original Problem (S,E,F)

Level 1 Design

Logical Structure (S_1,E_1,F_1)

Level 2 Design

Physical Structure (S_2,E_2,F_2)

Level 3 Design

Detailed Design (s,e,f)

Figure 6-7 Three-level Design

At each level the design follows the process of Section 6.2.2. The following is a detailed description.

6.3.2.1 Level 1 Design: Original Problem to Logical Structure

At this level, we focus on the functional requirements of the design problem, i.e., E_i and F_i do not involve any physical variables. The physical constraints are ignored for the time being. The goal is to arrive at a design which specifies the task architecture, the authority architecture, the information structure and the communication structure at the logical level. This provides a design which is natural to the problem. It is, however, not necessarily compatible with the physical constraints. This will be taken care of at the next level. Of the four steps of the design process at this level, we shall concentrate our discussion on the first step.

a. Synthesis

As discussed before, synthesis is a creative step that requires good knowledge of the problem. It was also pointed out that a good representation of the problem will be extremely useful. The original problem under consideration is a decision problem, which can be viewed as a mapping of the measurement data from the external world into actions. This problem can be decomposed into smaller subproblems. One decomposition is separation of the task into situation assessment, then planning and control. Situation assessment can be further decomposed into hypothesis formation, evaluation and selection. Planning can be decomposed into option generation, evaluation and selection. If the data and actions can be partitioned, the partitioning can induce further decompositions on the various functions.

As a result of this decomposition, we obtain a task structure which is a useful representation for synthesizing logical designs. The task structure contains not only the tasks that need to be performed to accomplish the goals, but also their precedence relations, information requirements, response requirements and possible conflicts. Task 1 is a precedent of task 2 if task 1 has to be completed before task 2 can be undertaken. This also implies that information needs to flow from task 1 to task 2. The information requirements of the task structure can be displayed conveniently by means of a graph where each node corresponds to an input and/or output for a task in the decision

106

making process. A directed branch connects node x_i to node y_j if the input x_i is needed in generating the output y_j. The task structure will thus have a graph such as that of Figure 6-8. (x_1,x_2,x_3) and (y_1,y_2,y_3) are the original inputs and outputs in the decision process; (w_1,w_2,w_3) are additional input-output pairs which have been introduced in the task decomposition.

The logical structure can be synthesized in the following way. Each logical decision node is defined by its inputs and outputs. For example we can have four logical nodes for the task structure in Figure 6-8.

Node a: Inputs x_1,x_2 Outputs w_1,w_2

Node b: Input x_3 Output w_3

Node c: Inputs w_1,w_2 Outputs y_1,y_2

Node d: Input w_3 Output y_3

The graph of the task structure induces a natural communication structure in the logical structure. Since w_2 depends on x_3 and w_3 depends on x_1, bidirectional branches are induced between nodes a and b, i.e., they should communicate. Likewise, no communication is needed between nodes c and d since y_1 depends only on w_1. The logical structure is shown in Figure 6-9.

The task structure is just one way of representing the original problem to facilitate the synthesis of the logical structure. In general, synthesis depends on what we choose to represent the problem. If human decision makers are involved, a model similar to that described in Section 7 on organizational decision making can be used. This leads to a different logical structure with features such as authority structures which are always present in organizations.

A large number of logical structures can be synthesized for a given problem. To guide the synthesis, design rules based on experience and analysis should be used. Some of the candidate rules are:

Figure 6-8 Graphical Representation of Task Structure

Figure 6-9 Logical Structure

108

- Sensors and effectors needed to accomplish a given task should be assigned to the same logical node

- Tasks which are completely independent should be assigned to separate nodes

- Similar tasks should be assigned to the same node to reduce the amount of processing

- Tasks which have the same information requirements may be grouped together to minimize communication

- Important tasks should perhaps be assigned to more than one logical node to reduce vulnerability

- If conflicting tasks are assigned to multiple nodes, a higher level node may be needed for conflict resolution

- Low-level tasks, such as signal processing, should be assigned to the same nodes as the sensors

- If fast response time is needed, the path from the data to the response should be short, i.e., not too many nodes should be involved.

b. Optimization

Any free parameters can be optimized with respect to one criterion such as response time, survivability, etc.

c. Evaluation

The logical structure should be evaluated with respect to the requirements of the original problem. This may involve a simulation. However, if the decision nodes have not been specified in great detail, simple calculations may suffice.

d. Selection

If a number of logical structures are found to be acceptable, selection of one logical structure can be done using any of the methods described in Section 6.2.2. In particular, one should consider the promise that a logical

structure will lead to a satisfactory final design when other constraints are incorporated.

6.3.2.2 Level 2 Design: Logical Structure to Physical Structure

At this level, we look for designs which incorporate the physical constraints of the problem. These include the actual description of the sensors, effectors and decision makers. The resulting design (S_2, E_2, F_2) will specify the physical task architecture, authority architecture, information structure and communication structure. This design should be one which satisfies all the physical constraints and is natural to the problem under consideration. The same four step procedure discussed before is used.

a. Synthesis

A physical structure can be obtained by assigning the functions of the logical structure to the physical nodes. A natural way is to anchor the input and output variables of the logical structure to the physical sensors and effectors. After this, the logical decision nodes can then be assigned to physical decision nodes which are already present or to new physical decision nodes. The logical structure then induces the natural physical structure, i.e., task and authority architectures, information and communication structures. If there are additional constraints on these, such as limited communication between physical nodes, they should also be used in the synthesis. Sometimes a logical structure may not lead to an acceptable physical structure. For example, a communication path as specified by the logical structure may not be allowable by the physical constraints. In this case, it is necessary to regenerate a different logical structure. In general, even though level 1 is primarily concerned with the functional requirements of the problem, it is a good idea to look ahead and consider the kind of physical constraints one may have to consider in the future.

Design rules again would be useful in generating candidate designs. For survivability, a logical node which performs critical functions should be mapped into more than one physical node. If communication constraints are severe, logical nodes which communicate a lot should be placed in the same physical node.

110

b. Optimization

Various optimization problems can be formulated to fix the parameters in the design. For example, one can vary the locations of the physical decision nodes to minimize the communication requirements.

c. Evaluation

Simulation is usually the tool to check whether the design requirements, both physical and functional, can be satisfied.

d. Selection

At this time, the attributes of the design alternatives are usually known. Selection can be performed in many ways as discussed before.

6.3.2.3 Level 3 Design: Physical Structure to Detailed Design

With the selection of the physical structure, the basic design of the distributed decision making system is specified. If desired, one can go on to design the components which comprise the design. These would include the design of each decision node as well as the communication network.

a. Design of Decision Node

Each physical decision node is specified by its decision task, as well as the constraints on information and communication. The node may contain a human decision maker or its function may be automated by means of a computer. If a human decision maker is present, he may be assisted by means of a decision support system. Once the requirements of this system are specified, it can be designed using the methodology of Section 6.2. If a computer is used for automatic decision making, the hardware and software specifications can be obtained in the same way.

b. Design of Communication Network

The design of the actual communication network follows the physical intermodal communication requirements. If these requirements are known (from the physical structure) the network can be designed in the usual way [154].

The design of the distributed decision making system is now complete. Before the actual implementation of such a system, it may be necessary to do a full-scale simulation.

6.4 DESIGN EXAMPLE

In this section we consider a simple example to illustrate the design process. The system to be designed is for cruise missile offense against enemy land targets. The cruise missiles are to be launched from air platforms. Data about the enemy targets are collected by satellite based sensors. For survivability, a distributed decision making system is desired.

6.4.1 Design Problem

The design problem is specified by (S,E,F). S is the distributed decision making system transforming sensor data into command signals for the cruise missiles. E, the environment, consists of the enemy targets and defenses, the satellite sensors, the cruise missiles and the launch platforms. F gives the goals and performance specification, including survivability, performance in destroying enemy targets and cost. The desired final design (s,e,f) consists of s, the components of the distributed decision making system, e, the environment in which they are defined and f, the requirements on the components.

6.4.2 Design Process

We now use the steps described in Section 6.3 to go from the initial problem to the final design.

6.4.2.1 Level 1 Design: Logical Structure

The first step at this level is to generate some candidate designs. As discussed before, a proper representation of the decision problem is very useful. The decision problem can be viewed as the transformation of sensor data into command signals for the cruise missiles. A functional decomposition leads to the tasks shown in Figure 6-10. The decomposition can also be represented graphically as in Figure 6-11. To generate distributed logical structures, the inputs and outputs are partitioned by sensors, missiles and targets, e.g.,

112

SENSOR DATA

```
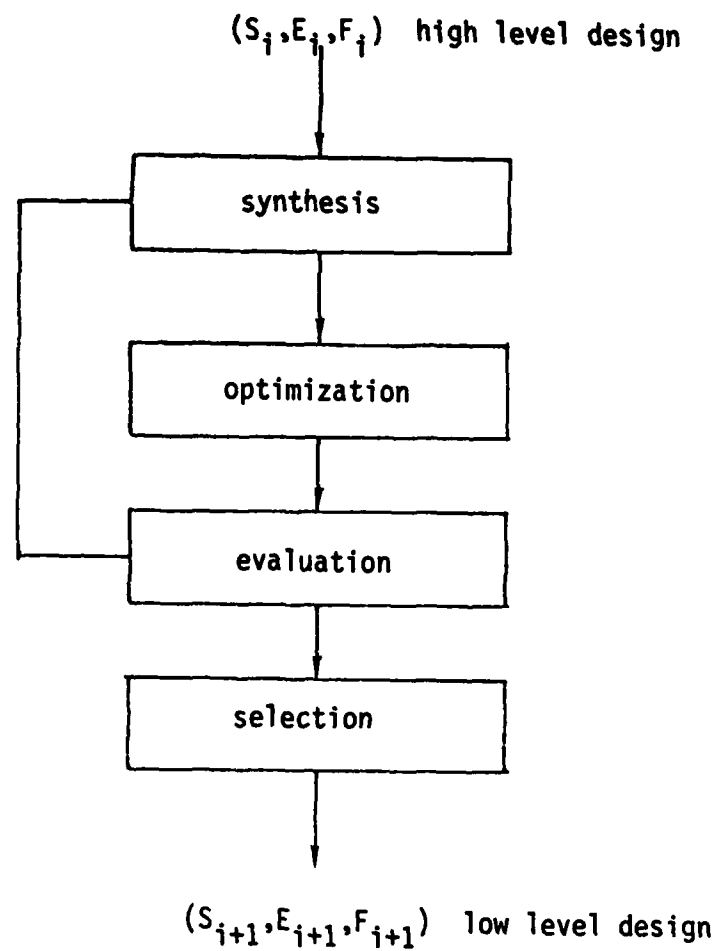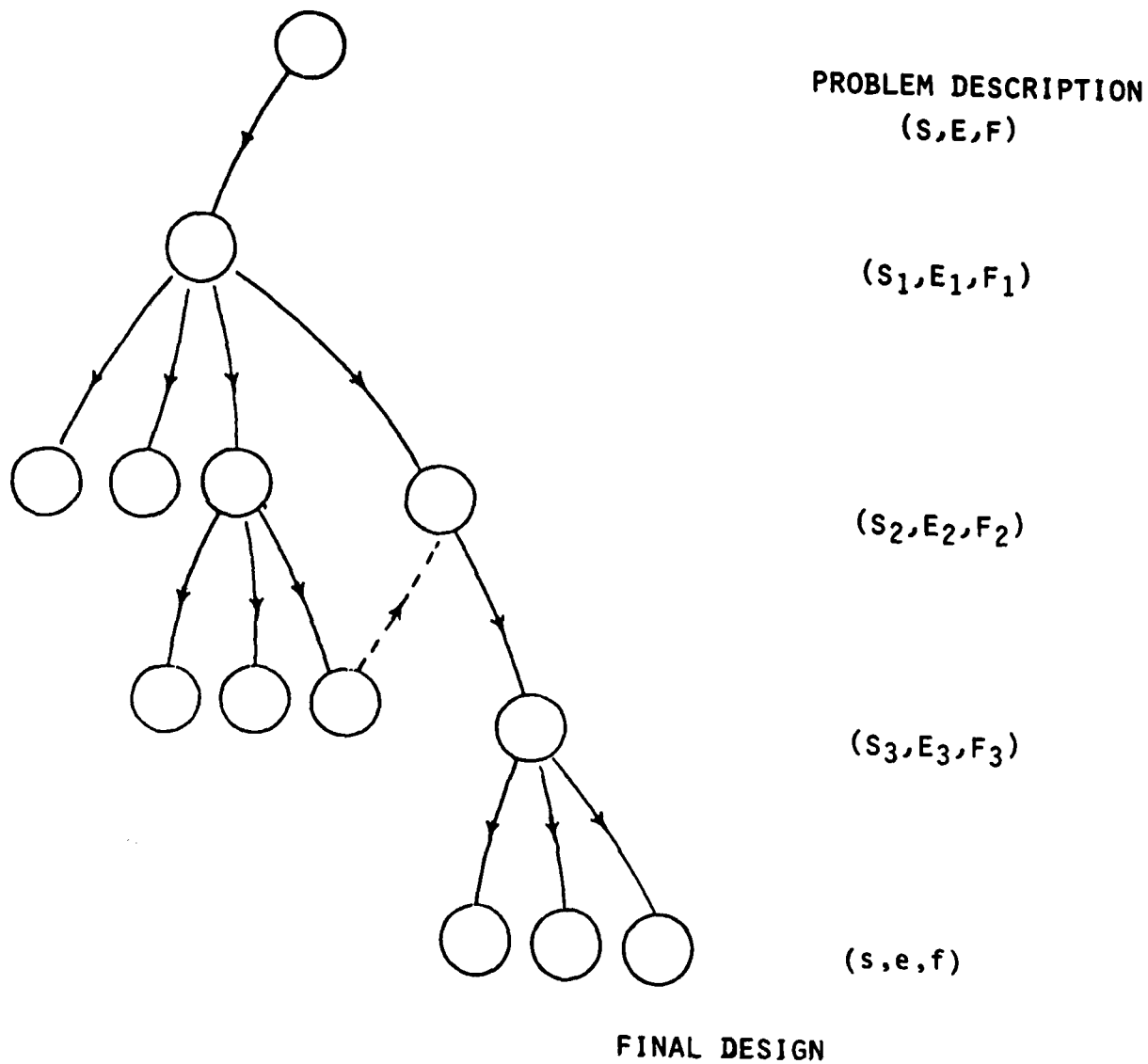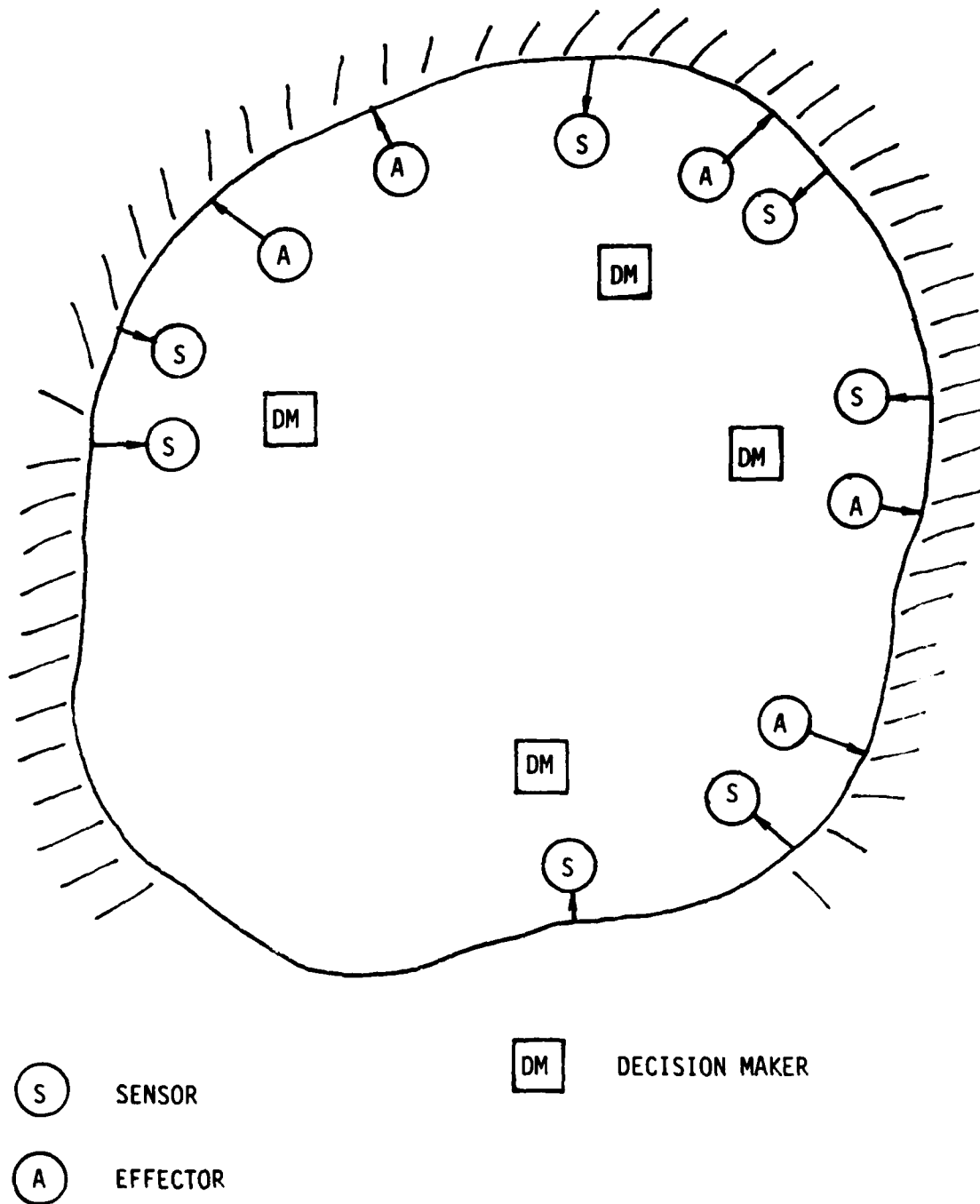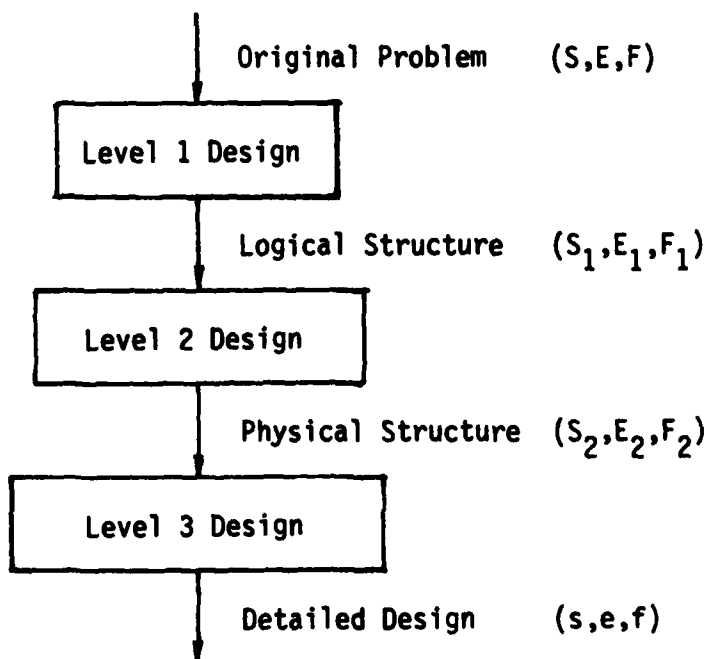        ┌─────────────────────────────┐
        │    SITUATION ASSESSMENT     │
        └─────────────────────────────┘
                      │
                      ▼
        ┌─────────────────────────────┐
        │       MISSION PLANNING      │
        └─────────────────────────────┘
                      │
                      ▼
        ┌─────────────────────────────┐
        │      PATH OPTIMIZATION      │
        └─────────────────────────────┘
                      │
                      ▼
        ┌─────────────────────────────┐
        │    GUIDANCE AND CONTROL     │
        └─────────────────────────────┘
                      │
                      ▼
```

CONTROLS FOR MISSILES

Figure 6-10 Functional Decomposition

```
   V        W        X        Y        Z
   ●───▶────●───▶────●───▶────●───▶────●
```

V: SENSOR DATA

W: STATUS OF MISSILES, TARGETS

X: ASSIGNMENT OF MISSILES TO TARGETS

Y: OPTIMAL PATHS FOR MISSILES

Z: CONTROLS FOR MISSILES

Figure 6-11 Graphical Representation

113

$$v = (v_1, \ldots \ldots, v_{NS})$$

$$w = (w_1, \ldots \ldots, w_{NM+NT})$$

$$x = (x_1, \ldots \ldots, x_{NM})$$

$$y = (y_1, \ldots \ldots, y_{NM})$$

$$z = (z_1, \ldots \ldots, z_{NM})$$

Where   NS = number of sensors

NM = number of missiles

NT = number of targets

Since the functions of path optimization and guidance and control are independent for different missiles, the task structure will have the form shown in Figure 6-12. This basically reflects that given a particular assignment of a missile M_i (x_i), the optimal path (y_i) and the controls for missile M_i (z_i) can be found independently of the other missiles.

V	W	X	Y	Z

SITUATION   PLANNING   PATH   GUIDANCE
ASSESSMENT   OPTIMIZATION   & CONTROL

Figure 6-12   Task Structure

114

The task structure represents the ultimate distributed logical design where each output node corresponds to a decision node. It is obvious that path optimization and guidance and control can be performed in a distributed manner. Thus from now on we focus on the design of logical structures for situation assessment and planning. By grouping together inputs and outputs, we arrive at two designs shown in Figure 6-13. Note that we have eliminated the totally centralized version as being undesirable.

Each logical decision node in the figure is responsible for the assignment of a missile M_i. In Figure 6-13(a), the same status assessment is available to all the decision nodes. In Figure 6-13(b), only partial information w^i = status of {all targets, missile M_i} is available for node i. In this case, communication is induced between all the decision nodes from the original task structure.

The requirements F_i may correspond to the reliability of the logical decision nodes. In Figure 6-13(a) the node generating w has to be quite reliable because all the other nodes depend on its information. The reliability requirements on the nodes w^i in Figure 6-13(b) are not that high. These requirements can be optimized given the overall requirements of the system as specified by F. Computational requirements are also obvious. Figure 6-13(a) implies the same computation duplicated at all the logical decision nodes. In Figure 6-13(b), the computation is distributed.

With the proper specifications, evaluation of the two designs show that they are both acceptable. We can either pick one to be designed at the next level or retain both at the same time. Our decision is to consider both of these alternatives.

6.4.2.2 Level 2 Design: Physical Structure

The sensor data come from the satellite sensors. Guidance and control should reside with each cruise missile. These are the easy assignments. Path optimization may reside in the cruise missile or another physical decision node. We shall concentrate on the assignment of w and x.

Assume that the cruise missiles are launched from two launch platforms. We also require that situation assessment and planning be assigned to the same

115

(a) Design I - Distributed with Centralized Information

(b) Design II - Distributed with Decentralized Information

Figure 6-13 Logical Structures

116

decision node since these are closely coupled functions. Then each of the two logical designs in Figure 6-13 gives rise to two physical designs as shown in Figure 6-14. In design IA, the decision node in charge of generating w and x_1 is a cruise missile. In design IB, there are two decision nodes, each generating w and the plans for a group of cruise missiles. In both of these designs, the physical decision nodes share the same data and carry out essentially the same computations. No communication between the decision nodes is needed.

Designs IIA and IIB are derived from logical design II. In IIA, each cruise missile is a physical decision node. In IIB, the launch platforms are the decision nodes. In both of these designs, the physical nodes have decentralized information and communicate with each other to arrive at a plan for assigning the missiles.

Still two more designs can be obtained by taking IIA and IIB and removing the communication. As expected, the performance of the overall system will suffer.

These designs are then evaluated with respect to communication and computation requirements as well as performance. One possible performance measure is the expected military value of the destroyed targets.

$$J = \sum_{i=1}^{NT} V_i P(T_i)$$

where NT is the number of targets

V_i is the military value of the ith target

$P(T_i)$ is the probability the ith target T_i is destroyed

For a set of 8 targets and 8 missiles with given physical parameters, the physical structures are compared in Table 6-1.

If we require J to be high, then designs IIIA and IIIB will be eliminated. IIB is also poor with respect to reliability. Thus the designs IA, IB and IIA are acceptable designs. If response time is critical, IIA is undesirable since it requires iterations between decision nodes. Also,

117

Figure 6-14   Physical Structures

118

Table 6-1 Comparison of Different Physical Architectures

	DATA BASE	COMPUTATION AT DECISION NODE	COMMUNICATION BETWEEN DECISION NODES	J NO FAILURE	J FAILURE OF L_1 OR L_2
Ia	Common	Duplicated among missiles	None	69	69
Ib	Common	Duplicated among launch platforms	None	69	69
IIa	Decentralized	Decentralized but iterative	Yes	69	69
IIb	Decentralized	Decentralized but iterative	Yes	69	30 or 28
IIIa	Decentralized	Decentralized	None	50	50
IIIb	Decentralized	Decentralized	None	58	30 or 28

J = Expected military value

frequent communication allows easy detection by the enemy. Thus IA and IB are acceptable designs.

If we compare IA and IB, both require the same computation at each node. On the basis of cost, IB is more desirable. However, IA is probably more reliable. A higher level criterion would be needed for the final selection.

6.4.2.3 Level 3 Design: Detailed Design

Both hardware specifications (computers, communication links) and software algorithms (for situation assessment, planning, path optimization, guidance and control, etc.) can now be investigated. At this stage, a detailed simulation is possible.

7. THE ROLE OF THE HUMAN DECISION MAKER

7.1 INTRODUCTION

Military decision making is an example of organizational decision making where the overall activities of the organization are a result of decisions and actions made by different organizational members. For this class of distributed decision making problems, where there are humans involved in assessment, generation of options, evaluation and choice; we need to examine the problem more closely in order to identify the important issues involved. Before doing so, we need to fix some definitional concepts to facilitate discussion.

A decision is considered as an irrevocable allocation of resources, and a decision maker is an individual who has the authority to commit the resources of an organization. In a distributed decision making environment, the authority over resource commitment is distributed in the sense that each decision maker has the authority to commit a portion of the total resources of an organization. Let R be the set of total resources of an organization, and let $N=\{1,...,n\}$ be the set of decision makers. The ith decision maker has the authority to commit resources $R_i \subset R$. It is reasonable to assume that

$$\bigcup_{i=1}^{n} R_i = R$$

and in some cases, we may also have

$$R_i \cap R_j = \emptyset \qquad \forall \, i \neq j$$

In the case where $R_i \cap R_j \neq 0$, then there are certain resources which can be allocated by either decision maker i or decision maker j and thus bargaining between these two decision makers may be necessary to determine the allocation of such resource. In this chapter, we shall discuss the issues involved, which then lead to identification of important problems in a

121

distributed decision making environment in the context of organizational
decision making.

7.2. DECISION MAKING PROCESS

Before discussing the issues involved in a human distributed decision
making environment, we shall first describe the decision making process of any
particular decision maker i within the organization. (See Figure 7-1) The
inputs to the decision making process are:

1. Set of guidelines or goal, and

2. Assessment of local situation.

The first phase is the issues identification process. In this process,
the decision maker assesses what his/her local situation is relative to the
overall situation and decides whether new reallocation is needed to fulfill
his own objective subject to the set of guidelines or goal stipulated (usually
by someone higher up in the authority structure); and if new reallocation is
needed, what issues need to be considered. As indicated in Figure 7-1,
assessment of the overall situation requires information which resides with
other decision makers.

The issues identified in this phase are inputted into the options
generation phase where options are created concerning how the resources
available to him/her should be reallocated. In the case where $R_i \cap R_j \neq 0$, this
phase is interrelated among different decision makers who are in control of
the same resources.

Each option generated is evaluated carefully by each decision maker
independently. In the evaluation, the individual's risk profile and critics'
comments are incorporated. Basically, decision maker i wants to assess to
what degree is his/her goal being fulfilled within the stipulated guidelines,
given that a certain option is chosen. The evaluation process may trigger new
options to be generated, either by identifying the weaknesses of the present
options generated or identifying new issues to be considered.

If an option is generated in which all decision makers agree upon the use
of the common resource, then the outcome is a specific distributed decision.
This is achieved by a bargaining process among decision makers who are in

122

SET OF   LOCAL
GUIDELINES   ASSESSMENT
OR "SYSTEM"
GOAL

INDIVIDUAL
GOAL ─────────────→ ┌──────────────────────┐
                     │   │   QUERY OTHER DMS
           ─────────→│ ISSUE IDENTIFICATION │←───→ TO ASSESS WHETHER
                     │   │   INDIVIDUAL GOAL IS
                     └──────────────────────┘   ACHIEVED VIA STATUS QUO
                              │ ISSUES TO
                              │ FOCUS ON
                              ↓
RESOURCE   ┌──────────────────────┐   NEGOTIATION WITH OTHER DMS;
UNDER CONTROL ─────────→ │   │←─→ PROPOSAL ON USE OF COMMON
                ───────→ │ OPTIONS GENERATION   │   RESOURCE
                         └──────────────────────┘
                              │
                              ↓
INDIVIDUAL
GOAL ─────────────→ ┌──────────────────────┐   QUERY OTHER DMS TO
RISK   │   │   ASSESS IMPACT OF OPTION
PROFILE ─────────────│   EVALUATION   │←───→ ON EVERYBODY; THIS IS
                     │   │   NEEDED FOR BARGAINING ON
CRITICS ─────────────→└──────────────────────┘   THE USE OF COMMON RESOURCE
                              │
                              ↓
                         ┌──────────────────────┐
                         │   │
                         │   CHOICE PROCESS   │
                         │   │
                         └──────────────────────┘
                              │ DECISION
                              ↓
                         ┌──────────────────────┐
                         │   │
                         │   IMPLEMENTATION   │
                         │   │
                         └──────────────────────┘
                              │ PLAN
                              ↓
                     ┌──────────────────────────┐
                     │ PROVIDE PROPER GUIDELINE │
                     │ (INCENTIVE) TO ACHIEVE PLAN│
                     └──────────────────────────┘
                      ↓ ↓ ↓ ↓ ↓ ↓ ↓

GUIDELINES TO ACTORS IN THE LOWER HIERARCHICAL LEVEL

Figure 7-1 Distributed Decision Making Process

123

control of the same resources. To implement the decision, a distributed plan can be developed which provides guidelines for decision makers (or actors) in the lower level of hierarchy. In order that the lower level actions will adhere to the desired plan, a proper guideline (or incentive) may be provided.

7.3 PROBLEM ISSUES

We can now identify certain problem issues in a distributed human decision making environment. We can categorize them into the following classes of subproblems.

7.3.1 Distributed Assessment and Evaluation

The problem here is to assess whether one's goal is fulfilled within the specified guideline when a certain option (or status quo) is implemented. There are four stages involved.

(i) "What is" -- This is a distributed hypothesis formation problem. The problem can be abstracted as follows. Each DM has his/her local information which allows him/her to construct independently his/her "local picture." This is a combination of model building and hypothesis testing. However, in order to see whether the global situation will influence his/her goal achievement, each DM will have to get information from other DMs. Since communication capacity is usually limited, each DM has to determine what information is needed from which DM and what query is to be sent to each DM. In response, each DM has to decide what information (type and aggregation level) is to be sent to other DMs. In general, query response is carried out in a sequential manner. Note that each DM is not interested in the overall situation, but only to the extent that relates to his/her individual goal achievement.

(ii) "What if" -- the "what is" gives a structural model of the situation, with certain parameters and variables which are either uncertain or nonstationary over time. In order to evaluate what would be the likely outcome conditioned on a chosen option, one needs to use the structural model to answer a set of "what if" questions. Plausible scenarios are to be constructed which are represented by setting certain values for the parameters and variables within the model, then using the

124

model, determine the plausible outcomes. Assessment of the likelihood of each plausible scenario will yield assessment on the likelihood of plausible outcomes.

(iii) Goal Representation -- The problem here is to model the individual goal in a quantitative way. A standard procedure is to determine a set of attributes, where the numerical values of the attributes represent how one perceives the relative "distance" between one's position and one's goal. Some of these attributes are physical in nature and their values can be measured absolutely (e.g., number of enemy targets destroyed), others are more abstract and only relative comparison is meaningful (e.g., the military value of destroying a specific enemy target). Note that a goal can have many different representations. An individual's risk profile must be incorporated in each representation. The process of utility encoding in any decision analysis practice is one way to determine one specific goal representation. The multi-goal representation has important implications and shall be discussed further in the choice process.

(iv) Performance Evaluation -- This is a straightforward problem once the model structure (what is) is determined, plausible outcomes (what if) are determined and different goal representations are specified. All that is required is to map the plausible outcomes to the set of attributes in each representation, and weigh them appropriately with the likelihood of the plausible outcome.

7.3.2 Issue Identification

This is accomplished by identifying both the "weak" and "strong" points which lead to goal accomplishment. The problem can be posed as follows: Given a certain individual goal representation in terms of a set of attributes, and a certain assessment of one's strategic position (these are obtained via distributed assessment and evaluation as discussed above), on which subset of the attributes should one focus one's attention in generating alternatives? The process requires more creativity than analytical skill (even though some high level analysis is required).

125

7.3.3 Option Generation

The issue identification process identifies the issues to be focused in generating alternatives. The next process is to generate alternatives based on these issues. Again this is mostly a creative process even though domain knowledge and analytical skill can be very helpful. In some cases, we can formulate such a process via a set of mathematical optimization problems [155]. In other cases, an option prompting environment is needed to induce the creative process [156]. In a distributed decision making environment, an option initiated by an individual decision maker must also include a proposal on how the common resources are to be allocated [157].

7.3.4 Choice Process

This requires explicitly bargaining among individual decision makers, at the same hierarchical level, on the use of common resources. Since each of them has a different goal and a different assessment (both local and global), the problem is a cooperative game with incomplete information. Some modifications of the classical formulation of such a game problem is required since goal representation is not uniquely defined.

7.3.5 Implementation, Organizational Structure and Incentive Setting

This is the phase where a decision is translated into actions to be taken. In an organization with hierarchical structure, a decision at a certain level needs to be translated into actions taken by decision makers at a lower level. The problem of implementation is the setting of guidelines for lower level actors such that their combined actions align with the decision made. Since each actor has his/her own individual goal, the problems of organizational structuring (number of actors, delegation of authority, communication channels, etc.) and incentive setting (influence goal representations) are extremely important. Note that for a lower level, we have another distributed decision making process as given in Figure 7-1. This phase requires much more psychological understanding of individual and group response than analytical skill, though simple modeling and analysis are helpful in certain situations where logical deduction is required.

126

8. CONCLUSIONS AND FUTURE EFFORT

We have surveyed literature relevant to distributed decision making. Through this survey, we learned some techniques that could be helpful for our DDM effort. We also learned that DDM is a large, complex problem for which there is no unified theory. We hope that this project will contribute to overall understanding of the problem.

A major thrust of the project has been the development of our interactive decision making environment. Our strategy has been to concentrate first on designing and implementing the software needed to support a single decision maker. After several rounds of improvement, we feel that this software has almost all of the capabilities that it should. After implementing a few more enhancements, the software should be to the point where we can begin to implement a distributed version of it, after which we will begin experimentation. We feel that much has been learned already from developing and using the existing software and that much more will be learned in our future efforts.

A related effort has been the development of automated decision making techniques. Some of these are still in the development stage while other. have been partially or fully implemented. In general, the more mathematica. approaches seem best suited for simplified problems or for lower level subproblems in a hierarchically decomposed, complex problem. In order to automate decision making for large, real-world problems, heuristics seem necessary. The most promising approach seems to be to use algorithms for the computationally intense aspects of a problem in conjunction with human or artificial intelligence to make more abstract decisions. Future efforts include further algorithm development, incorporating these algorithms into our interactive simulation, and investigating the feasibility of developing an AI planning system.

We have developed a methodology for designing distributed decision making systems. By a careful examination of the general design process and specializing it to our problem, we have identified the basic steps needed in the design process. We believe that although the design methodology cannot be automated at this point, a systematic approach is indeed available. The success or failure of the design depends very much on the experience and

127

knowledge of the designer. Tools, however, can be developed to assist him in the various tasks. The process is thus an interactive one with the human being playing the dominant role and assisted by a computer aided design support system.

We have also gained some preliminary insight into the effects of introducing humans into the decision making process. This issue will be investigated more fully when our distributed interactive planning environment has been implemented, as will the issues of distributed data bases and architectures.

References

[1] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton University Press, 1944.

[2] R. A. Fisher, "The fiducial argument in statistical inference," *Annals of Eugenics*, vol. 6, pp. 391-398, 1935.

[3] A. Wald, *Sequential Analysis*. New York, NY: Wiley, 1947

[4] L. J. Savage, *The Foundations of Statistics*. New York, NY: Wiley, 1954

[5] W. Edwards, "The theory of decision making," *Psychological Bulletin*, vol. 51, pp. 380-417, 1954.

[6] C. H. Coombs and S. S. Kornorita, "Measuring utility of money through decision," *American Journal of Psychology*, vol. 71, pp. 383-389, 1958.

[7] W. K. Estes, "Of models and men," *The American Psychologist*, vol. 12, no. 10, pp. 609-617, 1957.

[8] S. Siegel, "Decision making and learning under varying conditions of reinforcement," *Annals of the New York Academy of Science*, vol. 89, pp. 766-783, 1961.

[9] P. Slovic, B. Fischhoff and S. Lichtenstein, "Behavioral decision theory," *Annual Review of Psychology*, vol. 28, pp. 1-39, 1977

[10] H. J. Einhorn and R. M. Hogarth, "Behavioral decision theory: processes of judgment and choice," *Annual Review of Psychology*, vol. 32, pp. 53-88, 1981

[11] T. S. Wallstan (ed.). *Cognitive Processes in Choice and Behavior*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1980.

[12] P. Varaiya, "Trends in the theory of decision making in large systems," *Annals of Economics and Social Measurement*, vol. 1, no. 4, 1972.

[13] N. R. Sandell, Jr., P. Varaiya, M. Athans, and M. G. Safonov, "Survey of decentralized control methods for large scale systems," *IEEE Trans. Automat. Contr.*, vol. AC-23, no. 2, pp. 108-128, Apr. 1978.

[14] R. E. Larson, *Tutorial: Distributed Control*, IEEE Computer Society, 1979.

[15] C. B. McGuire and R. Radner (eds.). *Decision and Organization*. Amsterdam: North Holland, 1972

[16] A. M. Geoffrion, "Elements of large-scale mathematical programming: Part I: concepts; part II, synthesis of algorithms and bibliography," *Management Science*, vol. 16, no. 11, pp. 652-675, pp. 626-691, July, 1970.

[17] L. S. Lasdon, *Optimization Theory for Large Systems*. New York: MacMillan, 1970.

[18] D. A. Wismer, _Optimization Methods for Large-Scale Systems_, New York, NY: McGraw-Hill, 1971.

[19] B. Chandrasekaran (ed.). "Special Issue on Distributed Problem-Solving," _IEEE Trans. Syst., Man, Cybern._, vol. SMC-11, no. 1, pp. 1-99, Jan. 1981.

[20] R. D. Luce and H. Raiffa, _Games and Decisions: Introduction and Critical Survey_. New york, NY: Wiley, 1957.

[21] H. Raiffa and R. Schlaifer, _Applied Statistical Decision Theory_, Boston, MA: Graduate School of Business, Harvard University, 1961.

[22] J. W. Pratt, H. Raiffa, and R. Schlaifer, "The foundations of decision under uncertainty: an elementary exposition," _J. Am. Statis. Assoc._, vol. 59, pp. 353-375, June, 1964

[23] J. W. Pratt, H. Raiffa, and R. Schlaifer, _Introduction to Statistical Decision Theory_. New york, NY: McGraw-Hill, 1965.

[24] H. Raiffa, _Decision Analysis: Introductory Lectures or Choices under Uncertainty_, Reading, MA: Addison-Wesley, 1970.

[25] D. W. North, "A tutorial introduction to decision theory." _IEEE Trans. Systems and Cybernetics_, vol. SSC-4, no. 3, pp. 200-210, Sept. 1968.

[26] P. C. Fishburn, "Utility theory," _Management Sci._, vol. 14, pp. 335-378, Jan. 1968.

[27] R. A. Howard, "Decision Analysis: applied decision theory," _Proc. 4th Int. Conf. Operational Res._, Boston, MA, 1966.

[28] R. A. Howard, "The foundations of decision analysis," _IEEE Trans. Systems and Cybernetics_, vol. SSC-4, no. 3, pp. 1-9, Sept. 1968.

[29] R. A. Howard, J. E. Matheson and K. E. Miller (eds.). _Readings in Decision Analysis_. Menlo Park, CA: SRI International, 1977.

[30] S. Barclay, et al., "Handbook for decision analysis," Technical Report TR-77-6-30, Decisions and Design, Inc., 1977.

[31] R. A. Howard, J. E. Matheson and D. W. North, "The decision to seed hurricanes," _Science_, vol. 176, pp. 1191-1202, 1972.

[32] Stanford Research Institute, "Decision analysis of nuclear plants in electrical system expansion," SRI Project 6496 Final Report, 1968.

[33] R. L. Keeney and H. Raiffa, _Decisions with Multiple Objectives: Preferences and Value Tradeoffs_, New York, NY: Wiley, 1976.

[34] J. Marschak, "Elements of a theory of teams," _Management Sci._, vol. 1, pp. 127-137, 1955.

[35] T. Groves, "Incentives in teams," _Econometrica_, vol. 41, pp. 617-631, July 1973.

[36] K. J. Arrow and R. Radner, "Allocation of resources in large teams," _Econometrica_, vol. 47, no. 2, pp. 361-385, Mar. 1979.

[37] T. Groves and M. Loeb, "Incentives in a divisionalized firm," _Management Sci._, vol. 25, no. 3, pp. 221-230, 1979.

[38] R. Radner, "Team decision problems," _Ann. Math. Stat._, vol. 33, pp. 857-881, 1962.

[39] J. Marschak and R. Radner, _The Economic Theory of Teams_, Cowles Foundation Mono. 22. New Haven, Conn.: Yale University Press, 1972.

[40] M. Beckman, "Decision and team problems in airline reservations," _Econometrica_, vol. 26, pp. 134-145, 1958.

[41] Y. C. Ho and K. C. Chu, "Information structure in dynamic multi-person control problems," _Automatica_, vol. 10, pp. 341-351, 1974.

[42] Y. C. Ho, "Team decision theory and information structures," _Proc. IEEE_, vol. 68, no. 6, pp. 644-654, July, 1980.

[43] H. S. Witsenhausen, "On information structures, feedback and causality," _SIAM. J. Control_, vol. 9, no. 2, pp. 149-160, May, 1971.

[44] J. M. Bismut, "An example of interaction between information and control: the transparency of a game," _IEEE Trans. Automat. Contr._, vol. AC-18, no. 5, pp. 518-522, Oct. 1978.

[45] N. R. Sandell, Jr. and M. Athans, "Solutions of some nonclassical LQG stochastic decision problems," _IEEE Trans. Automat. Contr._, vol. AC-19, no. 2, pp. 108-116, Apr. 1974.

[46] Y. C. Ho, M. P. Kastner and E. Wong, "Teams, signaling, and information theory," _IEEE Trans. Automat. Contr._ vol. AC-23, no. 2, pp. 305-312, Apr. 1978.

[47] G. B. Dantzig and P. Wolfe, "The decomposition algorithm for linear program," _Econometrica_, vol. 29, pp. 767-778, 1961.

[48] K. J. Arrow and L. Hurwicz, "Decentralization and computation in resource allocation," in _Essays in Economics and Econometrics_, R. W. Pfouts, Ed. Chapel Hill, NC: Univ. of North Carolina Press, 1960, pp. 34-104.

[49] L. S. Lasdon and J. D. Schoeffer, "A multilevel technique for optimization," _Proc. JACC_, Troy, N.Y. 1965.

[50] G. Cohen, "Optimization by decomposition and coordination: a unified approach," _IEEE Trans. Automat. Contr._, vol. AC-23, no. 2, pp. 222-232, Apr. 1978.

[51] M. R. Javdan and R. J. Richards, "Decentralized control systems theory: a critical evaluation," _Int. J. Control_, vol. 26, no. 1, pp. 129-144, 1977.

[52]   R. Bellman, *Dynamic Programming*, Princeton, NJ: Princeton Univ. Press,
        1957.

[53]   C. Y. Chong, P. L. McEntire, and R. E. Larson, "Decomposition of
        mathematical programming by dynamic programming," *Proc. 2nd Lawrence
        Symp. Syst. and Decision Sci.*, Oct. 1978.

[54]   T. B. Cline and R. E. Larson, "Decision and control in large-scale
        systems via spatial dynamic programming," *Proc. 1st Lawrence Symp.
        Syst. and Decision Sci.*, Oct. 1977.

[55]   P. L. McEntire, C. Y. Chong and R. E. Larson, "A global optimality
        theorem for spatial dynamic programming," *Proc. 2nd Lawrence Symp.
        Syst. and Decision Sci.*, Oct. 1978.

[56]   B. Friedlander, "A decentralized strategy for resource allocation,"
        IEEE Trans. Automat. Contr. vol. AC-27, no. 1, pp. 260-265, Feb. 1982.

[57]   D. P. Bertsekas, "Distributed dynamic programming," *Proc. 20th IEEE
        Conf. on Decision and Control*, 1981.

[58]   J. M. McQuillan, G. Falk and I. Richer, "A review of the development
        and performance of the ARPANET routing algorithm," *IEEE Trans.
        on Comm.*, vol. COM-26, no. 12, pp. 1802-1810, Dec. 1978.

[59]   R. Lau, R. C. M. Persiano and P. P. Varaiya, "Decentralized information
        and control: a network flow example," *IEEE Trans. Automat. Contr.*,
        vol. AC-17, no. 4, pp. 466-473, Aug. 1972.

[60]   A. Gersho and B. J. Karafin, "Mutual synchronization of geographically
        separated oscillators," *Bell Syst. Tech. J.*, vol. 45, no. 10, pp.
        1689-1704, 1966.

[61]   I. W. Sandberg, "On conditions under which it is possible to synchronize
        digital transmission systems," *Bell Syst. Tech. J.*, vol. 48, no. 6,
        pp. 1999-2020, 1969.

[62]   D. G. Cantor and M. Gerla, "Optimal routing in a packet-switched computer
        network," *IEEE Trans. on Computers*, vol. C-23, no. 10, pp. 1062-1069,
        Oct. 1974.

[63]   F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C.
        Walden, "The interface message processor for the ARPA Computer
        Network," *Conf. Rec., Spring Joint Comput. Conf.*, AFIPS Conf.
        Proc., 1970, pp. 551-566.

[64]   J. McQuillan, I. Richer and E. Rosen, "The new routing algorithm for
        the ARPANET," *IEEE Trans. Commun.*, vol. COM-28, pp. 711-719, 1980.

[65]   J. M. Abram, "Some shortest-path algorithms with decentralized informa-
        tion and communication requirements," Doctoral Dissertation,
        Washington University, St. Louis, 1981.

[66] J. M. Abram and I. B. Rhodes, "Some shortest-path algorithms with decentralized information and communication requirements," to appear in IEEE Trans. Automat. Contr.

[67] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," IEEE Trans. on Comm., vol. COM-27, no. 9, pp. 1280-1287, Sept. 1979.

[68] R. E. Kahn, "Resource-sharing computer communication networks," Proc. IEEE, vol. 60, pp. 1397-1407, Nov. 1972.

[69] M. Schwartz and T. E. Stern, "Routing techniques used in computer communication networks," IEEE Trans. Commun., vol. COM-28, pp. 539-552, Apr. 1980

[70] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," IEEE Trans. on Comm., vol. COM-25, no. 1, pp. 73-85, Jan. 1977.

[71] D. P. Bertsekas, E. Gafni and K. S. Vastola, "Validation of algorithms for optimal routing of flows in networks," Proc. 17th IEEE Conf. on Decision and Control, 1978.

[72] D. P. Bertsekas, "Algorithms for optimal routing of flow in networks," Coordinated Science Lab., Working Paper, Univ. Illinois, Urbana, ILL, June, 1978.

[73] J. S. Meditch and J. C. Mandojana, "A decentralized algorithm for optimal routing in data-communication networks," Proc. 18th IEEE Conf. on Decision and Control, 1979.

[74] T. E. Stern, "A class of decentralized routing algorithms using relaxation," IEEE Trans. Commun., vol. COM-25, 1, pp. 1092-10, Oct. 1977.

[75] A. Segall, "The modeling of adaptive routing in data-communication networks," IEEE Trans. on Comm., vol. COM-25, no. 1, pp. 85-95, Jan. 1977.

[76] J. S. Meditch and F. D. Gorecki, "Minimum hop flow assignment and routing in computer-communication networks," Proc. 19th IEEE Conf. on Decision and Control, 1980.

[77] J. S. Meditch and F. D. Gorecki, "A distributed minimum hop routing algorithm," Proc. 20th IEEE Conf. on Decision and Control, 1981.

[78] C. M. Heal, "Planning without prices," Rev. Econ. Studies, vol. 36, pp. 347-362, 1963.

[79] Y. C. Ho, L. Servi and R. Suri, "A class of center-free resource allocation algorithms," J. Large-Scale Syst., vol. 1, Feb. 1980.

[80] L. D. Servi, "Electrical networks and resource allocation algorithms," IEEE Trans. Syst., Man., Cybern., vol. SMC-10, no. 12, pp. 841-848, Dec. 1980.

[81] K. J. Astrom, _Introduction to Stochastic Control Theory_. New York, NY: Academic Press, 1970.

[82] H. S. Witsenhausen, "Separation of estimation and control for discrete time systems," _Proc. IEEE_, vol. 59, pp. 1557-1566, 1971.

[83] T. Yoshikawa, "Dynamic programming approach to decentralized stochastic control problems." _IEEE Trans. Automat. Contr._, vol. AC-20, no. 6 pp. 796-797, Ec. 1975.

[84] H. S. Witsenhausen, "A counterexample in stochastic optimum control," _SIAM Journ. Control_, vol. 6, pp. 131-147, 1968.

[85] I. B. Rhodes and D. G. Luenberger, "Stochastic differential games with constrained state estimators," _IEEE Trans. Automat. Contr._, vol. AC-14, no. 5, pp. 476-481, Oct. 1969.

[86] W. Willman, "Formal solutions for a class of stochastic pursuit-evasion games," _IEEE Trans. Automat. Contr._, vol. AC-14, no. 5, pp. 504-509, Oct., 1969.

[87] C. Y. Chong and M. Athans, "On the stochastic control of linear systems with different information sets," _IEEE Trans. Automat. Contr._, vol. AC-16, no.5, pp. 423-430, Oct. 1971.

[88] A. Segall and N. R. Sandell, Jr., "Dynamic file assignment in a computer network--part II, decentralized control," _IEEE Trans. Automat. Contr._, vol. AC-24, no. 5, pp. 709-715, Oct. 1979.

[89] F. C. Schoute, "Symmetric team problems and multi access wire communication," _Automatica_, vol. 14, pp. 255-269, 1978.

[90] F. C. Schoute, "Decentralized control in packet switched satellite communication," _IEEE Trans. Automat. Contr._, vol. AC-23, no. 2, pp. 362-371, Apr. 1978.

[91] P. Varaiya and J. Walrand, "Decentralized control in packet switched satellite communication," _IEEE Trans. Automat. Contr._, vol. AC-24 no. 5, pp. 794-796, Oct. 1979.

[92] B. Hajek, "Dynamic decentralized estimation and control in a multi-access broadcast channel," _Proc. 19th IEEE Conf. on Decision and Control_, 1980.

[93] J. W. Grizzle, S. I. Marcus and K. Hsu, "Decentralized control of a multiaccess broadcast network," _Proc. 20th IEEE Conf. on Decision and Control_, 1981.

[94] M. D. Meserovic, D. Macko and Y. Takahara, _Theory of Hierarchical Multilevel Systems_, New York, NY: Academic Press, 1970.

[95]   M. G. Singh, _Dynamical Hierarchical Control_, Amsterdam, The Netherlands:
       North-Holland, 1977.

[96]   M. G. Singh and A. Titli, _Systems: Decomposition, Optimization and
       Control_, Oxford: Pergamon, 1978

[97]   W. Findeisen, F. N. Bailey, M. Brdys, K. Malinowski, P. Tatjewoki and
       A. Wozniak, _Control and Coordination in Hierarchical Systems_,
       New York, NY: Wiley, 1980.

[98]   M. S. Mahmoud, "Multilevel systems control and application: a survey,"
       IEEE Trans. Syst., Man and Cybern., vol. SMC-7, no. 3, pp. 125-143.
       Mar. 1977.

[99]   W. Findeisen, et al., "On-line hierarchical control for steady-state
       systems," _IEEE Trans. Automat. Conts._, vol. AC-23, no. 2, pp. 189-209,
       Apr. 1978

[100] C. Y. Chong and M. Athans, "On the periodic coordination of linear
       stochastic systems," _Automatica_, vol. 12, July 1976.

[101] J. B. Cruz, Jr., "Decentralized multicriteria optimization of linear
       stochastic systems," _IEEE Trans. Automat. Contr._, vol. AC-23, no. 2,
       pp. 244-255, Apr. 1978.

[102] T. Basar and H. Selbuz, "Closed-loop Stackelberg strategies with applica-
       tions in the optimal control of multilevel systems," _IEEE Trans. Automat.
       Contr._, vol. AC-24, no. 2, pp. 166-179, Apr. 1979.

[103] C. Y. Cheng, "Some notions of decentralization and coordination in
       large-scale dynamic systems," _Large-Scale Dynamic Systems - A Seminar
       Workshop_ held at Utah State University, Logan, Utah, 1974, NASA-SP371

[104] R. R. Tenney and N. R. Sandell, Jr., "Structures for distributed decision
       making," _IEEE Trans. Syst., Man., Cybern._, vol. SMC-11, no. 8, pp.
       517-527, Aug. 1981.

[105] R. R. Tenney and N. R. Sandell, Jr., "Strategies for distributed
       decision making," _IEEE Trans. Syst., Man., Cybern._, vol. SMC-11,
       no. 8, pp. 527-538, Aug. 1981.

[106] H. K. Khalil and P. V. Kokotovic, "Control strategies for decision
       makers using different models of the same system," _IEEE Trans.
       Automat. Contr._ vol. AC-23, no. 2, pp. 289-298, Apr. 1978.

[107] Nils J. Nilsson, _Principles of Artificial Intelligence_ (Tioga
       Publishing Company, 1980).

[108] Edward A. Feigenbaum, "The Art of Artificial Intelligence: Themes
       and Case Studies of Knowledge Engineering," _Proceedings_ IJCAI-77
       (1977).

[109]   Robert Lindsay, Bruce G. Buchanan, Edward A. Feigenbaum, and
        Joshua Lederberg, DENDRAL (1981).

[110]   Edward H. Shortliffe and Bruce G. Buchanan, "A Model of Inexact
        Reasoning in Medicine," Mathematical Biosciences, 23 (1975).

[111]   Richard M. Tong, "Some Properties of Fuzzy Feedback Systems,"
        IEEE Trans. on Systems, Man, and Cybernetics, SMC-10, 6 (1980).

[112]   Richard M. Tong and P.P. Bonissone, "A Linguistic Approach to Decision
        Making with Fuzzy Sets," IEEE Trans. on Systems, Man, and Cybernetics,
        SMC-10, 11 (1980).

[113]   Robert Wesson, Problem Solving with Simulation in the World of an
        Air Traffic Controller, Ph.D. thesis (University of Texas at Austin,
        1977).

[114]   Lee D. Erman and Victor R. Lesser, "A Multilevel Organization for
        Problem Solving Using Many Diverse Cooperating Sources of Knowledge,"
        Proceedings IJCAI-75 (1975).

[115]   Robert J. Drazovich and Scottie Brooks, "Surveillance Integration
        Automation Project (SIAP), "Distributed Sensor Nets: Proceedings
        of a Workshop (Carnegie-Mellon University, December 1978).

[116]   Cordell Green and Brian P. McCune, "Application of Knowledge Based
        Programming to Signal Understanding Systems," Distributed Sensor
        Nets: Proceedings of a Workshop (Carnegie-Mellon University,
        December 1978).

[117]   Douglas B. Lenat, "BEINGS: Knowledge as Interacting Experts,"
        Proceedings IJCAI-75 (1975).

[118]   Richard D. Fennell and Victor R. Lesser, "Parallelism in AI Problem
        Solving: A Case Study of HEARSAY-II," IEEE Transactions on
        Computers, C-26,2 (February 1977).

[119]   G.G. Hendrix, E.D. Sacerdoti, D. Sagalowicz, and J. Slocum,
        "Developing a Natural Language Interface to Complex Data,"
        ACM Transactions on Data Base Systems, 3, 2 (June 1978).

[120]   Randy Davis, "Report on the Workshop on Distributed AI," SIGART
        Newsletter, 73 (October 1980).

[121]   Reid G. Smith and Randall Davis, "Distributed Problem Solving:
        The Contract Net Approach," Proceedings of the 2nd National
        Conference of the Canadian Society for Computational Studies of
        Intelligence, Toronto, Canada, July 1978.

[122]   Reid G. Smith and Randall Davis, "Applications of the Contract
        Net Framework: Distributed Sensing," Distributed Sensor Nets:
        Proceedings of a Workshop (Carnegie-Mellon University, December 1978).

[123]   Information Processing Techniques Office, Defense Advanced
         Research Projects Agency, <u>Distributed Sensor Nets: Proceedings</u>
         <u>of a Workshop</u> (Carnegie-Mellon University, December 1978).

[124]   Victor R. Lesser and Daniel D. Corkill, "Functionally Accurate
         Distributed Problem Solving Systems," <u>Distributed Sensor Nets:</u>
         <u>Proceedings of a Workshop</u> (Carnegie-Mellon University, December
         1978).

[125]   Victor R. Lesser and Daniel D. Corkill, "Distributed Interpretation
         Testbed," in Report on the Second Workshop on Distributed AI, edited
         by Randy Davis, <u>SIGART Newsletter</u>, 80 (April 1982).

[126]   Victor R. Lesser and Daniel D. Corkill, "Functionally Accurate,
         Cooperative Distributed Systems," special issue on distributed
         problem solving, <u>IEEE Transactions on Systems, Man and Cybernetics</u>
         (1981).

[127]   Carl Hewitt and Bill Kornfeld, "Message Passing Semantics," <u>SIGART</u>
         <u>Newsletter</u>, 73 (October 1980).

[128]   R. Wesson and F. Hayes-Roth, "Network Structures for Distributed
         Situation Assessment," Report R-2560-ARPA (Rand Corporation,
         August 1980).

[129]   Victor R. Lesser and Lee D. Erman, "An Experiment in Distributed
         Interpretation," special issue on distributed processing, <u>IEEE</u>
         <u>Transactions on Computers</u> (December 1980).

[130]   "Working Papers in Distributed Computation," Distributed Communication
         Group, University of Massachusetts (June 1981).

[131]   R. Steeb, S. Commarata, F. Hayes-Roth, and R. Wesson, "Distributed
         Intelligence for Air Fleet Control," R-2728-ARPA (Rand Corporation,
         1981).

[132]   Randy Davis, "Models of Problem Solving: Why Cooperate?" <u>SIGART</u>
         <u>Newsletter</u>, 73 (October 1980).

[133]   Frederick Hayes-Roth, "Towards a Framework for Distributed AI,"
         <u>SIGART Newsletter</u>, 73 (October 1980).

[134]   Richard Fikes, "A Commitment-based Framework for Describing Informal
         Cooperative Work," Xerox Palo Alto Research Center (June 1981).

[135]   Carl Hewitt, "The APIARY Network Architecture for Knowledgeable
         Systems," <u>Proceedings</u>, LISP Conference (Stanford University, August
         1980).

[136]   Eugene Ball and Phil Hayes, "Representation of Task-Specific
         Knowledge in a Gracefully Interacting User Interface," <u>Proceedings</u>
         of the 1st Annual National Conference on Artificial Intelligence
         (Stanford University, August 1980).

[137]   D. Gorlin, F. Hayes-Roth, S. Rosenschein, and H. Sowizral, "The ROSIE Language Reference Manual," Technical Report (Rand Corporation, 1980).

[138]   J.A. Feldman and K.R. Sloan, Jr., "Progress at the Rochester Image Understanding Project," Proceedings: Image Understanding Workshop (University of Maryland, April 1980).

[139]   A. Rosenfeld, "Iterative Methods in Image Analysis," Pattern Recognition, 10 (1978).

[140]   Lee D. Erman and Victor R. Lesser, "A Multilevel Organization for Problem Solving Using Many Diverse Cooperating Sources of Knowledge," Proceedings IJCAI-75 (1975).

[141]   Richard P. Wishner, Robert J. Drazovich, and Chee-Yee Chong, "Distributed Hypothesis Formation in Distributed Sensor Networks," Proposal 8021 (Advanced Information & Decision Systems, April 1980).

[142]   Robert J. Drazovich, Brian P. McCune, and Bruce G. Buchanan, "Characteristics of Hypothesis Formation Systems," Technical Report (Advanced Information & Decision Systems, 1981) (in press).

[143]   Carl Hewitt "Viewing Control Structures as Patterns of Passing Messages," Artificial Intelligence 8 (1977), 323-364.

[144]   William A. Kornfield "The Use of Parallelism to Implement a Heuristic Search," A.I. Memo No. 627, MIT AI Laboratory (March 1981).

[145]   William A. Kornfield and Carl E. Hewitt, "The Scientific Community Metaphor," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-11, No-1 (January 1981).

[146]   James F. Allen and Steven L. Small "The Rochester Discourse Comprehension Project" in Report on the Second Workshop on Distributed AI, edited by Randy Davis, SIGART Newsletter, 80 (April 1982).

[147]   Randy Davis, Dan Brotsky, and Judy Zinnikas "Teamwork in Multi-Agent Planning; Distribution as an Approach to Complexity" in Report on the Second Workshop on Distributed AI, edited by Randy Davis, SIGART Newsletter, 80 (April 1982).

[148]   Kurt Konolige and Nils Nilsson, "Multiple Agent Planning Systems," Proceedings AAAI, Stanford, CA, 1980.

[149]   Douglas E. Appelt, "A Planner for Reasoning about Knowledge and Action," Proceedings of the First Annual National Conference on Artificial Intelligence, 1980.

[150]   James Allen and C.R. Perrault, "Participating in Dialogues: Understanding Via Plan Deduction," Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence, 1978.

[151]   Philip Cohen and C.R. Perrault, "Elements of a Plan Based Theory of Speech Acts," Cognitive Science, vol. 3, pp. 177-212, 1979.

[152]   C. Alexander, Notes on the Synthesis of Form, Cambridge, MA: Harvard University Press, 1964.

[153]   T.L. Saaty, The Analytic Hierarchy Process, New York, NY: McGraw-Hill, 1980.

[154]   M. Schwartz, Computer Communication Network Design and Analysis, Princeton, NJ: Prentice-Hall, 1977.

[155]   E. Tse, "An Interactive Approach to Dynamic Sequence Assignment," AI&DS Technical Memo (under preparation).

[156]   R.M. Tong, et. al., "Options Generation Techniques for Command and Control," Final Report for Project 3012, AI&DS, 1982.

[157]   S. Mori and E. Tse, "Theory of Bargaining: A Game Theoretical Approach," Proceedings 1982 American Control Conference, Arlington, VA.

Appendix

A DISTRIBUTED RESOURCE ALLOCATION
ALGORITHM BASED ON EXTENDED MARGINAL ANALYSIS

140

1. INTRODUCTION

Marginal analysis is commonly applied to resource allocation problems with separable return functions. However, in many applications, the return function is nonseparable, and thus the classical marginal analysis is not applicable. In this section, we shall extend the marginal analysis to a certain case where the return function is nonseparable. A set of necessary and sufficient conditions for optimal allocation is given. The analysis easily lends itself to distributed implementation.

2. PROBLEM STATEMENT

The allocation problem is given as follows:

$$
\left.
\begin{aligned}
&\max \quad J = \sum_{i=1}^{n} f_i(x_i; y_j, j \; S_i) \\
&\text{s.t.} \quad \sum_{i=1}^{n} x_i + \sum_{j=k}^{s} y_j = R \\
&\qquad x_i, y_j \in I_+ = \{0,1,2,\ldots\} \\
&\qquad S_i \subset \{1,\ldots,s\} \qquad ; i=1,\ldots,n
\end{aligned}
\right\} \quad (P)
$$

where f_i has the following properties:

(a) $f_i(0; y_j, j \in S_i) = 0$ for all $y_j \in I_+$

(b) f_i is nondecreasing in each component while holding the rest of the components fixed

(c) For all $i=1,\ldots,n$ $(x_i > 0, y_t > 0, y_{t'} > 0)$

$f_i(x_i+1; y_j, j \in S_i) + f_i(x_i-1; y_j, j \in S_i) \leq 2f_i(x_i; y_j, j \in S_i)$

$f_i(x_i+1; y_t-1, y_j, j \in S_i - t) + f_i(x_i-1; y_t+1, y_j, j \in S_i - t) \leq 2f_i(x_i; y_j, j \in S_i); \forall t \in S_i$

$f_i(x_i; y_t+1, y_j, j \in S_i - t) + f_i(x_i; y_t-1, y_j, j \in S_i - t) \leq 2f_i(x_i; y_j, j \in S_i) \forall t \in S_i$

$f_i(x_i; y_t+1, y_{t'}-1, j \in S_i - \{t,t'\}) + f_i(x_i; y_t-1, y_{t'}+1, j \in S_i - \{t,t'\})$

$\qquad \leq 2f_i(x_i; y_j, j \in S_i) \; \forall t, t' \in S_i$

Note that this corresponds to "concavity" of f_i.

141

Note that (P) is more general than the usual resource allocation problem where the return function is separable. One class of problems that has the above formulation is in mission planning where we are to allocate R resource units to n targets which are defended by s defense sites. S_i represents the subset of defense sites that can "protect" target i. The function $f_i(x_i;y_j,j\epsilon S_i)$ represents the probability of destroying the ith target, and its functional form implies that the success of destroying target i depends on (1) allocation of x_i units to target i and (2) allocation of y_j units to those defense sites which protect target i.

3. A TRADING MODEL

Let us define a concept of price which will be utilized in the later discussions. For a given allocation $\{x_i\},\{y_j\}$, define for $x_i > 0$

$$\lambda_i^-(x_i;y_j,j\epsilon S_i) \overset{\Delta}{=} f_i(x_i;y_j,j\epsilon S_i) - f_i(x_i-1;y_j,j\epsilon S_i)$$

and for $y_j>0$, $G_j \overset{\Delta}{=} \{i|j\epsilon S_i\}$, $y^j \overset{\Delta}{=} \{y_1,\ldots,y_{j-1},y_{j+1},\ldots,y_s\}$

$$\rho_j^-(y_j,y^j;x_i,i\epsilon G_j) \overset{\Delta}{=} \sum_{i\epsilon G_j} \{f_i(x_i;y_\ell,\ell\epsilon S_i) - f_i(x_i;y_j-1,y_\ell,\ell\epsilon S_i^{-j})\}$$

For completeness, we define $\lambda_i^-(0;\cdot) = \infty$, $\rho_j^-(0,y^j;\cdot) = \infty$.

Note that λ_i^- is equal to the marginal decrease (in absolute value) in return if one resource unit is taken away from the ith target from its nominal allocation. Now if we imagine that there are n agents, each of them controlling resource units $\{x_i\}$ assigned to the ith target, then for agent i to give up one resource unit, he must charge a price equal to the marginal decrease in return due to the reduction of one resource unit. Therefore, one can interpret λ_i^- as the "selling price" for one resource unit asked by agent i who controls x_i.

Similarly, if we imagine that there are s agents controlling the resource units $\{y_j\}$ assigned to the jth defense site, then ρ_j^- has the interpretation of "selling price" for one resource unit asked by the agent controlling y_j.

Analogous to the concept of selling price, we have a concept of "buying" price." Define

$$\lambda_i^+(x_i; y_j, j\epsilon S_i) = f_i(x_i + 1; y_j, j\epsilon S_i) - f_i(x_i; y_j, j\epsilon S_i)$$

This has the interpretation of the buying price that the agent controlling x_i is willing to pay for an additional resource unit from an outside source. Similarly, one can define

$$\rho_j^+(y_j, y^j; x_i, i\epsilon G_j) = \sum_{i\epsilon G_j} \{f_i(x_i; y_j+1, y_\ell, \ell\epsilon S_i-j) - f_i(x_i; y_\ell, \ell\epsilon S_i)\}$$

With the above notion of prices, we have easily the following properties:

(1) $\lambda_i^-(x_i; y_j, j\epsilon S_i) = \lambda_i^+(x_i - 1; y_j, j\epsilon S_i)$

(2) $\lambda_i^-(x_i; y_j, j\epsilon S_i) \geqslant \lambda_i^+(x_i; y_j, j\epsilon S_i)$ [from property (c)]

(3) $\rho_j^-(y_j, y^j; x_i, i\epsilon G_j) = \rho_j^+(y_j - 1, y^j; x_i, i\epsilon G_j)$

(4) $\rho_j^-(y_j, y^j; x_i, i \epsilon G_j) \geqslant \rho_j^+(y_j, y^j; x_i, i\epsilon G_j)$ [from (c)]

Properties (1) and (3) come from our definition and the implication of properties (2) and (4) is that at each agent, the selling price for one unit resource is greater than the buying price for one unit resource.

With the above concept, we can investigate the "trading" pattern among agents, where the result of a trading yields a better overall performance; i.e., J increases. Let us divide the agents into two groups: X-group and Y-group. The X-group controls the resource units allocated to the targets and the Y-group controls the resource units allocated to the defenses. We have the following trading patterns:

143

(1) agents in X-group trade among themselves

(2) agents in Y-group trade among themselves

(3) agents in X-group trade with agents in Y-group

We can also have hierarchical trading patterns as illustrated in Fig. 1.

Figure A-1 Hierarchical Trading Pattern

4. TRADING EQUILIBRIUM AND OPTIMAL ALLOCATION

In this section we shall investigate the above trading patterns, and
relate the concept of trading equilibrium to optimal allocation.

<u>Theorem 1</u>: Let $\{x_i\}, \{y_j\}$ be a given feasible allocation. Assume now trading
takes place among X-group with $\{y_j\}$ remaining the same. If there exist some i, i'
such that

$$\lambda_i^+(x_i; y_j, j\epsilon S_i) > \lambda_{i'}^-(x_{i'}; y_j, j\epsilon S_i), \qquad (1)$$

then trading will take place between agents i and i' where at least one
resource unit will be transferred from i' to i.

144

Proof: Equation (1) implies that

$$f_i(x_i + 1; y_j, j \epsilon S_i) - f_i(x_i; y_j, j \epsilon S_i) > f_{i'}(x_{i'}; y_j, j \epsilon S_{i'}) - f_{i'}(x_{i'} - 1; y_j, j \epsilon S_{i'})$$

and thus for the new allocation $\{x_i^o\}, \{y_j\}$ where

$$x_i^o = x_i + 1 \quad ; x_{i'}^o = x_{i'} - 1 \qquad x_t^o = x_t \quad t \neq i, i'$$

we have

$$\sum_{i=1}^{n} f_i(x_i^o; y_j, j \epsilon S_i) > \sum_{i=1}^{n} f_i(x_i; y_j, j \epsilon S_i)$$

and thus trading will take place between agents i and i' which results in a better overall performance.

Consider the situation where $\{y_j\}$ is held fixed, then trading among agents in group X continues as long as (1) is satisfied for some i, i'; and every time a unit resource transaction is completed, the overall performance is improved. The trading sequence will result in a final allocation where no more trading will take place. We shall call such an allocation an equilibrium allocation among the X-group while $\{y_j\}$ is held fixed.

Theorem 2: Let $\{y_j\}$ be given with $\sum_{j=1}^{s} y_j < R$. The equilibrium allocation among the X-group, $\{\hat{x}_i\}$, is characterized by

$$\lambda_i^+(\hat{x}_i; y_j, j \epsilon S_i) \leq \lambda_{i'}^-(\hat{x}_{i'}; y_j, j \epsilon S_{i'}) \qquad \begin{matrix} i = 1, \dots, n \\ i' = 1, \dots, n \\ i \neq i' \end{matrix} \qquad (2)$$

Moreover, $\{x_i\}$ solves the optimization problem

$$\left. \begin{aligned} \max \quad J &= \sum_{i=1}^{n} f_i(x_i; y_j, j \epsilon S_i) \\ \text{s.t.} \quad \sum_{i=1}^{n} x_i &= R - \sum_{j=1}^{s} y_j \quad ; x_i \epsilon I^+ \end{aligned} \right\} \quad P(y)$$

145

Proof: From theorem 1, we see that (2) must be satisfied if no trading occurs. All we need to establish is that if (2) is satisfied, then no trading occurs. Let us consider an allocation $\{x_i^o\}, \{y_i\}$ where

$$x_i^o = \hat{x}_i + k_i \quad ; \quad \sum_{i=1} k_i = 0$$

One can imagine $\{x_i^o\}$ as the resulting allocation, as deviated from $\{\hat{x}_i\}$, if a trade has occurred among agents in X-group who are having initial allocation $\{\hat{x}_i\}$. For any i, k_i can either be positive or negative. Suppose $k_i > 0$, then

$$f_i(x_i^o; y_j, j \epsilon S_i) = f_i(x_i + k_i; y_j, j \epsilon S_i)$$

$$= \sum_{\gamma=0}^{k_i-1} \lambda_i^+(x_i + \gamma; y_j, j \epsilon S_i) + f_i(\hat{x}_i; y_j, j \epsilon S_i) \qquad (3)$$

By the first inequality in assumption (c), we have

$$f_i(x_i; y_j, j \epsilon S_i) > \frac{1}{2} f_i(x_i+1; y_j, j \epsilon S_i) + \frac{1}{2} f_i(x_i-1; y_j, j \epsilon S_i)$$

or for all $x_i > 0$

$$\lambda_i^+(x_i-1; y_j, j \epsilon S_i) \geqslant \lambda_i^+(x_i; y_j, j \epsilon S_i)$$

which implies that $\lambda_i^+(x_i; \cdot)$ is decreasing in x_i. Using this fact, (3) becomes

$$f_i(x_i^o; y_j, j \epsilon S_i) \leqslant k_i \lambda_i^+(\hat{x}_i; y_j, j \epsilon S_i) + f_i(\hat{x}_i; y_j, j \epsilon S_i)$$

If $k_i < 0$, then using the similar argument, we have

$$f_i(x_i^o; y_j, j \epsilon S_i) = -\sum_{\gamma=0}^{|k_i|-1} \lambda_i^-(\hat{x}_i; ; y_j, j \epsilon S_i) + f_i(\hat{x}_i; y_j, j \epsilon S_i)$$

$$\leqslant k_i \lambda_i^-(\hat{x}_i; y_j, j \epsilon S_i) + f_i(\hat{x}_i; y_j, j \epsilon S_i) . \qquad (4)$$

Let $\Pi = \min_{i=1,\ldots n} \lambda_i^-(x_i; y_j, j \epsilon S_i)$, then (1),(3) and (4) gives

146

$$f_i(x_i^0; y_j, j \in S_i) \leqslant k_i \Pi + f_i(\hat{x}_i; y_j, j \in S_i) \tag{5}$$

since $\sum\limits_{i=1}^{n} k_i = 0$, thus

$$\sum_{i=1}^{n} f_i(x_i^0; y_j, j \in S_i) \leqslant \sum_{i=1}^{n} f_i(\hat{x}_i; y_j, j \in S_i) \tag{6}$$

and therefore no incentive for such trading to occur. Note also that (6) implies x_i is an optimal allocation for $P(y)$.

Next, we shall consider trading among agents in Y-group. We shall say that j and j' are _related_ if $G_j \cap G_{j'} \neq 0$; otherwise they are unrelated.

Theorem 3: Let $\{x_i\}, \{y_i\}$ be a feasible allocation. Assume trading takes place among agents in Y-group with $\{x_i\}$ remaining the same. If j, j' are related and if $(y^{(j,j')} = \{y_1, \ldots, y_s\} - \{y_j, y_{j'}\})$

$$\rho_j^+(y_j, y_{j'} - 1, y^{(j,j')}; x_i, i \in G_j) > \rho_{j'}^-(y_{j'}, y^{j'}; x_i, i \in G_j) \tag{7}$$

then trading occurs between j and j' where at least one resource unit is transferred from agent j' to agent j. On the other hand, if j, j' are unrelated, trading occurs by transferring one unit from j' to j if we have

$$\rho_j^+(y_j, y^j; x_i, i \in G_j) > \rho_{j'}^-(y_{j'}, y^{j'}; x_i, i \in G_{j'}) \tag{7'}$$

Proof: The unrelated case is similar to the case of trading in X-group and the proof is similiar to that for Theorem 1. Therefore we shall concentrate only on the related case.

147

Let us partition the set $I_n \triangleq \{1,2,\ldots n\}$ into $G_j \cap G_{j'}$, $G_j | G_{j'}$, $G_{j'} | G_j$ and $I_n | G_j \cup G_{j'}$.[†] If j, j' are related, $G_j \cap G_{j'}$ is nonempty. Consider a new allocation $\{y_j^0\}$ with

$$y_j^0 = y_j + 1 \qquad ; y_{j'}^0 = y_{j'} - 1 \qquad ; y_t^c = y_t \quad t \neq j, j'$$

For $i \epsilon G_j \cap G_{j'}$, we have

$$f_i(x_i; y_\ell^0, \ell \epsilon S_i) = f_i(x_i; y_j+1, y_{j'}-1, y_\ell, \ell \epsilon S_i - \{j,j'\})$$

$$= f_i(x_i; y_j+1, y_j-1, y_\ell, \ell \epsilon S_i - \{j,j'\})$$

$$- f_i(x_i; y_{j'}-1, y_\ell, \ell \epsilon S_i - j') + f_i(x; y_{j'}-1, y_\ell, \ell \epsilon S_i - j')$$

$$- f_i(x_i; y_j, j \epsilon S_i) + f_i(x_i; y_j, j \epsilon S_i) \tag{8}$$

for $i \epsilon G_j | G_{j'}$, we have

$$f_i(x_i; y_\ell^0, \ell \epsilon S_i) = f_i(x_i; y_j+1, y_\ell, \ell \epsilon S_i - j) - f(x_i; y_\ell, \ell \epsilon S_i)$$

$$+ f_i(x_i; y_\ell, \ell \epsilon S_i) \tag{9}$$

for $i \epsilon G_{j'} | G_j$, we have

$$f_i(x_i; y_\ell^0, \ell \epsilon S_i) = f_i(x_i; y_{j'}-1, y_\ell, \ell \epsilon S_i - j') - f_i(x_i; y_\ell, \ell \epsilon S_i)$$

$$+ f_i(x_i; y_\ell, \ell \epsilon S_i) \tag{10}$$

and for $i \epsilon I_n | G_j \cup G_{j'}$, we have

$$f_i(x_i; y_\ell^0, \ell \epsilon S_i) = f_i(x_i; y_\ell, \ell \epsilon S_i) \tag{11}$$

[†]$A | B = \{a \epsilon A | a \notin B\}$

148

Combining (8)-(11) and using the price definitions, we have

$$\sum_{i=1}^{n} f_i(x_i; y_\ell^o, \ell \epsilon S_i) = \rho_j^+(y_j, y_{j'}, -1, y^{(j,j')}; x_i, i \epsilon G_j)$$

$$- \rho_{j'}^-(y_{j'}, y^{j'}; x_i \epsilon G_{j'}) + \sum_{i=1}^{n} f_i(x_i; y_\ell, \ell \epsilon S_i) \quad (12)$$

therefore if (7) is true, trading will occur between j' and j which results in a better allocation.

We can define equilibrium allocation among Y-group as we did for X-group.

<u>Theorem 4</u>: Let $\{x_i\}$ be given with $\sum\limits_{i=1}^{n} x_i < R$. The equilibrium allocation among the Y-group, $\{\hat{y}_j\}$, is characterized by

$$\rho_j^+(\hat{y}_j, \hat{y}_{j'}, -1, \hat{y}^{(j,j')}; x_i, i \epsilon G_j) \leqslant \rho_{j'}^-(\hat{y}_{j'}, \hat{y}^{j'}; x_i, i \epsilon G_{j'}) \quad (13)$$

if j and j' are related, and

$$\rho_j^+(\hat{y}_j, \hat{y}^j; x_i, i \epsilon G_j) \leqslant \rho_{j'}^-(\hat{y}_{j'}, \hat{y}^{j'}; x_i, i \epsilon G_{j'}) \quad (13')$$

if j and j' are unrelated. Moreover $\{\hat{y}_j\}$ solves

$$\left. \begin{array}{l} \max \sum\limits_{i=1}^{n} f_i(x_i; y_j, j \epsilon S_i) \\ \\ \text{s.t.} \sum\limits_{j=1}^{s} y_j = R - \sum\limits_{i=1}^{n} x_i \qquad ; y_j \epsilon I^+ \end{array} \right\} \quad P(x)$$

Proof: The proof is similar to that for Theorem 2 while utilizing (8), (9),(10), (11) and the third and fourth inequalities in assumption (c); but algebrically is more complicated because of the "relationship" among the agents in Y-group.

Finally, we shall consider trading between agents in X and Y groups.
We shall say that an agent i in X-group and an agent j in Y-group are
dependent if $j \epsilon S_i$ (or equivalently $i \epsilon G_j$).

Theorem 5: Let $\{x_i\}, \{y_j\}$ be a feasible allocation. If i in X-group and
j in Y-group are dependent, and if

$$\rho_j^+(y_j, y^j, x_i-1, x_t, t \epsilon G_j-i) > \lambda_i^-(x_i; y_\ell, \ell \epsilon S_i) \qquad (14)$$

trading occurs between these two agents with at least one resource
unit being transferred from i to j; and if

$$\lambda_i^+(x_i; y_j-1, y_\ell, \ell \epsilon S_i-j) > \rho_j^-(y_j, y^j; x_t, t \epsilon G_j) \qquad (15)$$

then trading occurs with at least one resource unit being transferred
from j to i.

If, however   nd j are not dependent, then trading occurs with
at least one unit   g tr nsferred from i to j if

$$\rho_j^+(y_j \qquad \qquad) > \lambda_i^-(x; y_\ell, \ell \epsilon S_i) \qquad (14')$$

and one unit being   ed from j to i if

$$\lambda_i^+(x_i; y_\ell, \qquad) > \rho_j^-(y_j, y^j; x_t, t \epsilon G_j) \qquad (15')$$

Proof: This proof is similar to those for Theorem 1 and Theorem 3. In the proof,
the second inequality in assumption (c) is utilized.

We shall say that a certain allocation $\{x_i^*\}, \{y_j^*\}$ is in global equilibrium
if there is no incentive for any agents, either in X or Y group, to trade
with each other.

150

Theorem 6: A global equilibrium allocation $\{x_i^*\}, \{y_j^*\}$ is characterized by the following conditions

$$
\text{(C*)}\left\{
\begin{array}{ll}
(1) & \lambda_i^+(x_i^*; y_j^*, j \in S_i) \leq \lambda_{i'}^-(x_{i'}^*; y_j^*, j \in S_i); \quad \begin{array}{l} i' = 1, \ldots, n \\ i = 1, \ldots, n \end{array} \\[4mm]
(2) & \rho_j^+(y_j^*, y_{j'}^*, -1, y^{*(j,j')}; x_i^*, i \in G_j) \leq \rho_{j'}^-(y_{j'}^*, y^{*j'}; x_i^*, i \in G_j) \\[2mm]
& \qquad\qquad\qquad\qquad\qquad\qquad \text{for all } (j, j') \text{ which are related} \\[3mm]
& \rho_j^+(y_j^*, y^{*j}; x_i^*, i \in G_j) \leq \rho_{j'}^-(y_{j'}^*, y^{*j'}; x_i^*, i \in G_{j'}) \\[2mm]
& \qquad\qquad\qquad\qquad\qquad\quad \text{for all } (j, j') \text{ which are not related} \\[4mm]
(3) & \rho_j^+(y_j^*, y^{*j}; x_i^* - 1, x_t^*, t \in G_j - i) \leq \lambda_i^-(x_i^*; y_\ell^*, \ell \in S_i); \quad \forall i \in G_j, \; j = 1, \ldots, s \\[4mm]
& \rho_j^+(y_j^*, y^{*j}, x_t^*, t \in G_j) \leq \lambda_i^-(x_i^*; y_\ell^*, \ell \in S_i) \quad \forall i \notin G_j, \; j = 1, \ldots, s \\[4mm]
& \lambda_i^+(x_i^*; y_j^* - 1, y_\ell^*, \ell \in S_i - j) \leq \rho_j^-(y_j^*, y^{*j}; x_t^*, t \in G_j) \quad \forall j \in S_i, \; i = 1, \ldots, n \\[4mm]
& \lambda_i^+(x_i^*; y_\ell^*, \ell \in S_i) \leq \ell_j^-(y_j^*, y^{*j}; x_t^*, t \in G_j) \quad \forall j \notin S_i, \; i = 1, \ldots, n
\end{array}
\right.
$$

Moreover, if $\{x_i^*\}, \{y_j^*\}$ is a global equilibrium allocation, it also solves (P).

Proof: This comes directly from combination of Theorem 1 to Theorem 5.

C* is the necessary and sufficient condition for both the global equilibrium allocation for the trading problem and the optimal solution for the allocation problem (P). An interesting interpretation of C* is possible.

Let us interpret

$$\rho_j^+(y_j, y_j-1, y^{(j,j')}; x_i \epsilon G_j)$$ — buying price j will offer for one additional resource from a related agent j' in Y-group

$$\rho_j^+(y_j; x_i-1, x_t, x_t \epsilon G_j-i)$$ — buying price j will offer for one additional resource from a dependent agent i in group X

$$\lambda_i^+(x_i; y_j-1, y_\ell, \ell \epsilon S_i-j)$$ — buying price that agent i in X-group will offer for one additional resource from a dependent agent j in group G

Then each agent has a selling price for one unit of resource; but dependent on where he buys his resource, has different buying prices for one more unit of resource. The global equilibrium is achieved when all the buying prices each agent is willing to give is lower than all the selling prices offered by the agents.

5. A DISTRIBUTED ALGORITHM

In this section, we shall describe a distributed algorithm based on Theorem 6. The algorithm is based on a sequence of "distributed trading" which leads to the trading equilibrium. Each trading cycle consists of two phases:

Phase 1: information exchange

Phase 2: unit resource trading

The purpose of Phase 1 is for each trading agent to compute his selling price and his set of buying prices (from different agents) for additional resources; in Phase 2, trading is to be carried out among agents in a distributed manner such that J is increased.

Phase 1: Information Exchange

Let us assume that, in order to evaluate

$$\{f_i(x_i+\alpha; y_j+\beta, j \epsilon S_i)\} \quad \alpha = -1,0,1; \quad \beta = -1,0,1$$

information exchange between agents i in X-group and j in Y-group ($j \epsilon S_i$)

must be carried out; which will, in turn, determine

$$\lambda_i^+(x_i;y_j,j\epsilon S_i),\ \lambda_i^+(x_i;y_j-1,y_\ell,\ell\epsilon S_i),\ \forall j\epsilon S_i$$

and

$$\lambda_i^-(x_i;y_j,j\epsilon Si).$$

After

$$\{f_i(x_i+\alpha;y_j+\beta,j\epsilon S_i)\}\quad \alpha=-1,0,1;\ \beta=-1,0,1$$

are determined for all $i=1,\ldots n$; then

$$\rho_j^+(y_j,y^j;x_i,i\epsilon G_j),\quad \rho_j^+(y_j,y_j,-1,y^{(j,j')};x_i,i\epsilon G_j)$$

and

$$\rho_j^-(y_j,y^j;x^i,i\epsilon G_j)$$

are determined for each agent j in Y-group.

Phase 2: Unit Resource Trading

We shall distinguish between three trading patterns:

(a) Trading among X-group

(b) Trading among Y-group

(c) Trading between X and Y groups

To describe how trading is to be carried out, we need to specify the set
of agents that each agent is allowed to trade with. For trading pattern (a),
each agent in X-group is allowed to trade with any one or more agents in
X-group. For trading pattern (b), each agent in Y-group is allowed to
trade with all related agents and one or more unrelated agents in Y-group.
For trading pattern (c), each agent in X-group is allowed to trade with
all its dependent agents and one or more independent agents in Y-group.

From Phase 1, selling and buying (dependent on agent to buy from)
prices are computed; then depending on the trading pattern each agent
trades with the allowable trading agent for one unit of resource. The
trading is to be carried out in the following sequence.

(1) Each agent determines, from the set of allowable trading
agents, a subset of agents who offer a buying price higher
than the agent's selling price (denote this subset as a list
of buyers). An agent with a nonempty list of buyers is
denoted as a selling agent.

153

(2) Each selling agent goes through his list of buyers and offers to the highest "bidder" a unit of resource at his selling price.

(3) A "buyer" who receives multiple offers chooses to receive one unit of resource from the seller who has the lowest selling price.

(4) Those selling agents whose offers are not accepted will form new lists by deleting the buyer who offered the highest price but chose to buy from other selling agents; these will be the new set of selling agents.

(5) If the set of selling agents is empty, Phase 2 is terminated, otherwise go back to (2) and iterate.

With the imposed "concavity conditions", one can show that successively iterating between Phase 1 and 2 will yield a sequence of monotonic improving resource allocations that will converge to the optimal allocation.